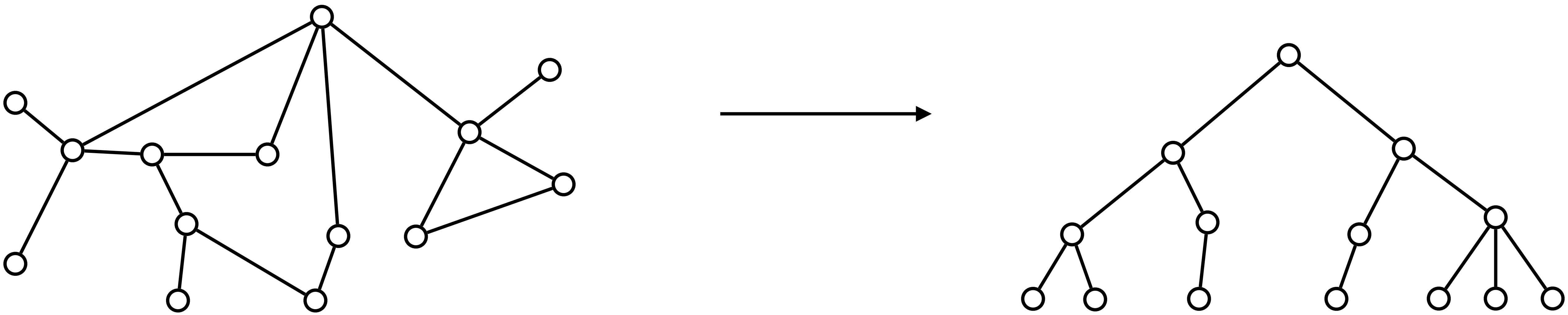
Online Network Design Algorithms via Hierarchical Decompositions

Appeared in SODA 2015

Seeun William Umboh

Eindhoven University of Technology

Overview



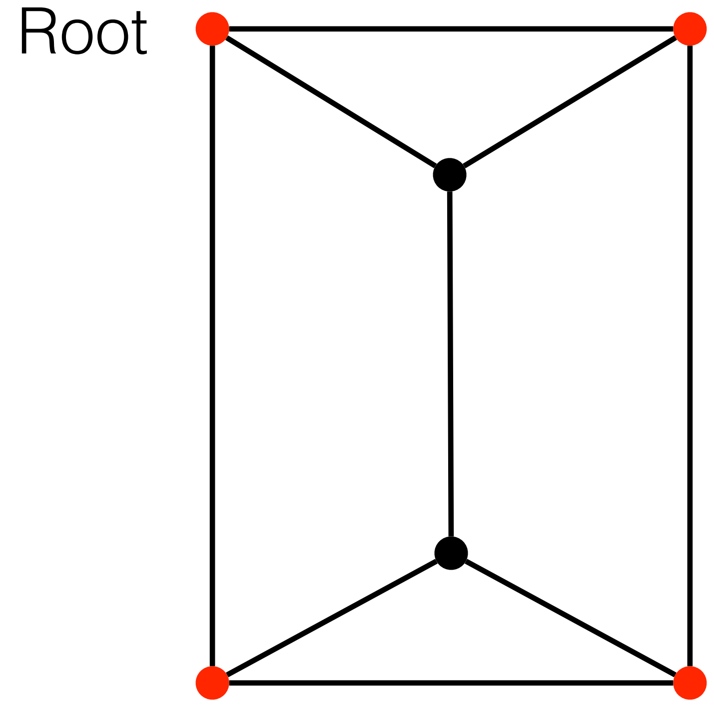
- Analysis framework based on tree embeddings
- Improved algorithms for online network design

Network Design

Fig. 7. INFONET communications networks.

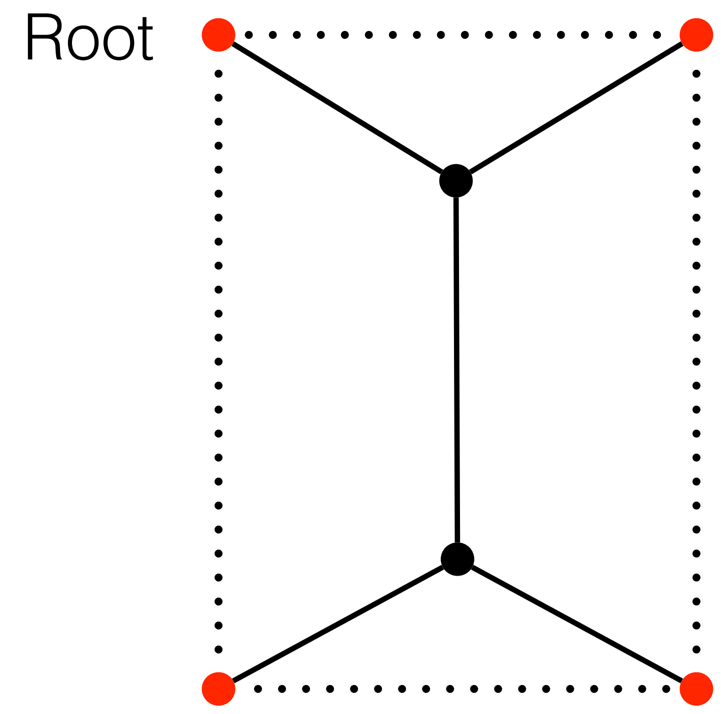
Given a graph G with edge costs and requirements, find a min-cost network in G

Steiner Tree



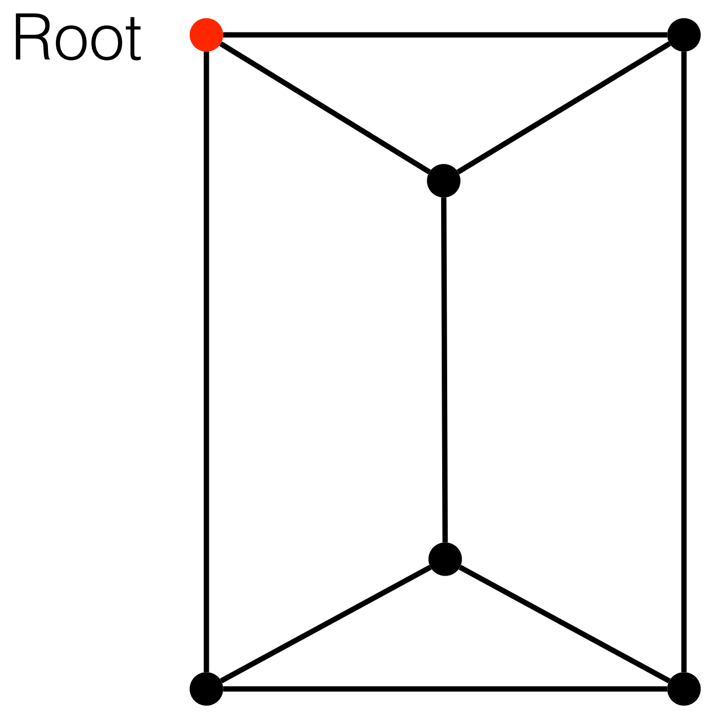
- Given graph and terminals
- Find min-cost subgraph connecting terminals

Steiner Tree



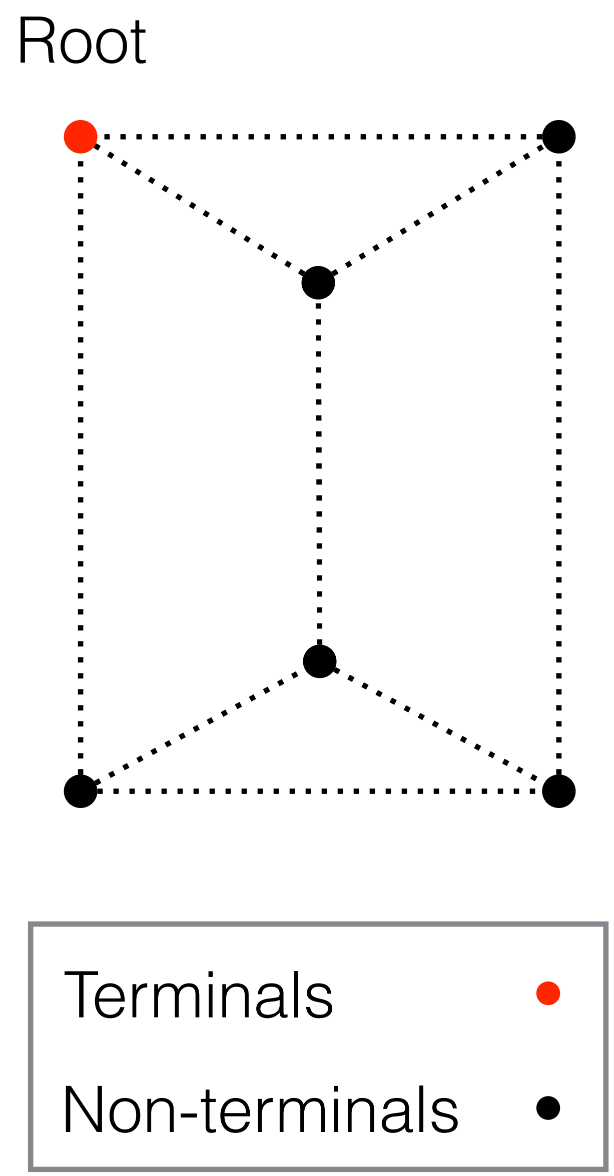
- Given graph and terminals
- Find min-cost subgraph connecting terminals

Online Steiner Tree

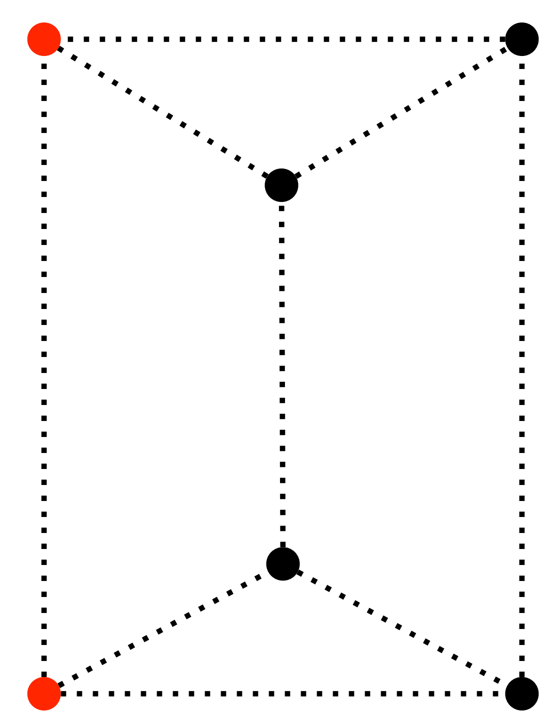


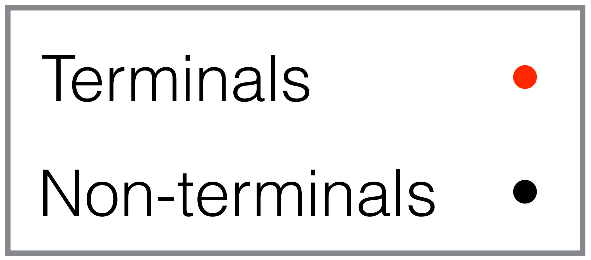
- Initially, only given graph and root
- At each time step, we are given a terminal i to connect
- Maintain min-cost subgraph connecting terminals to root
- Chosen edges cannot be removed later

Example

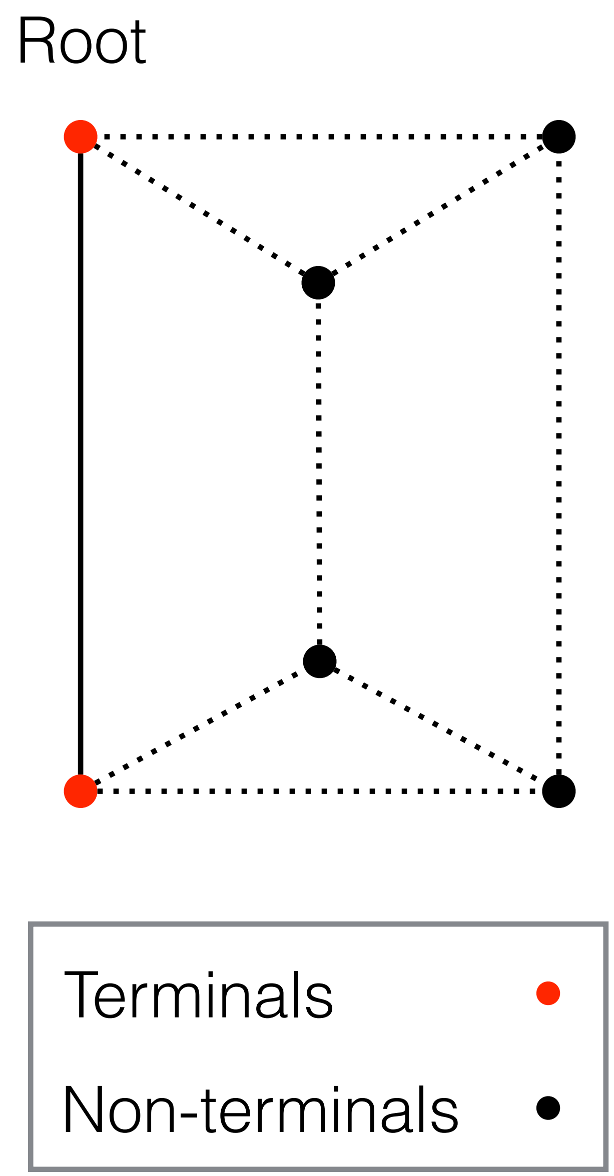


Example

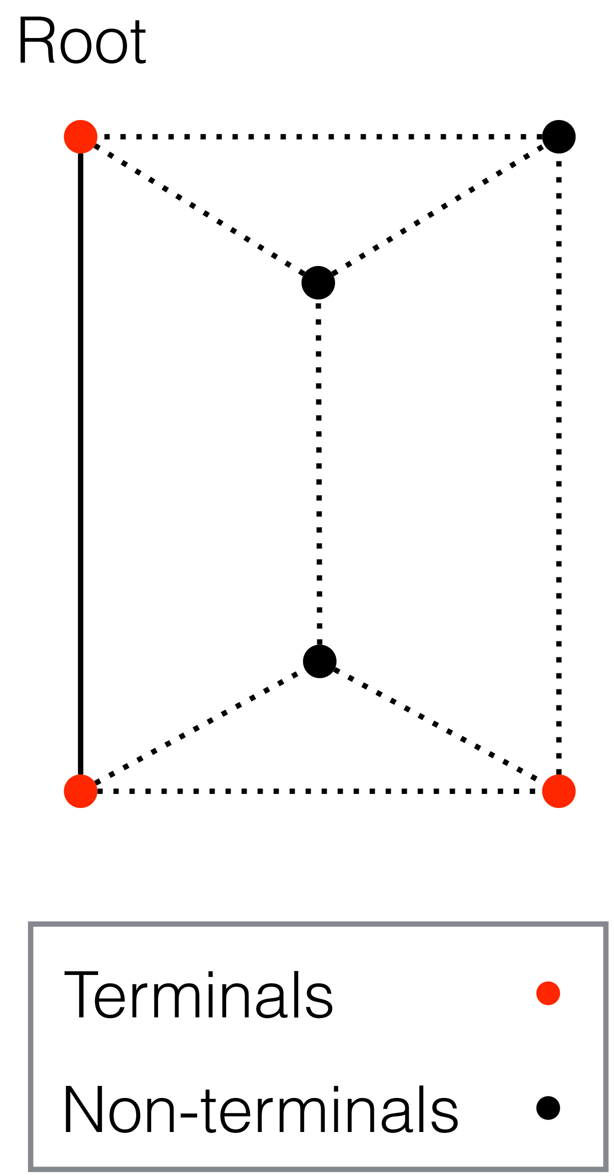




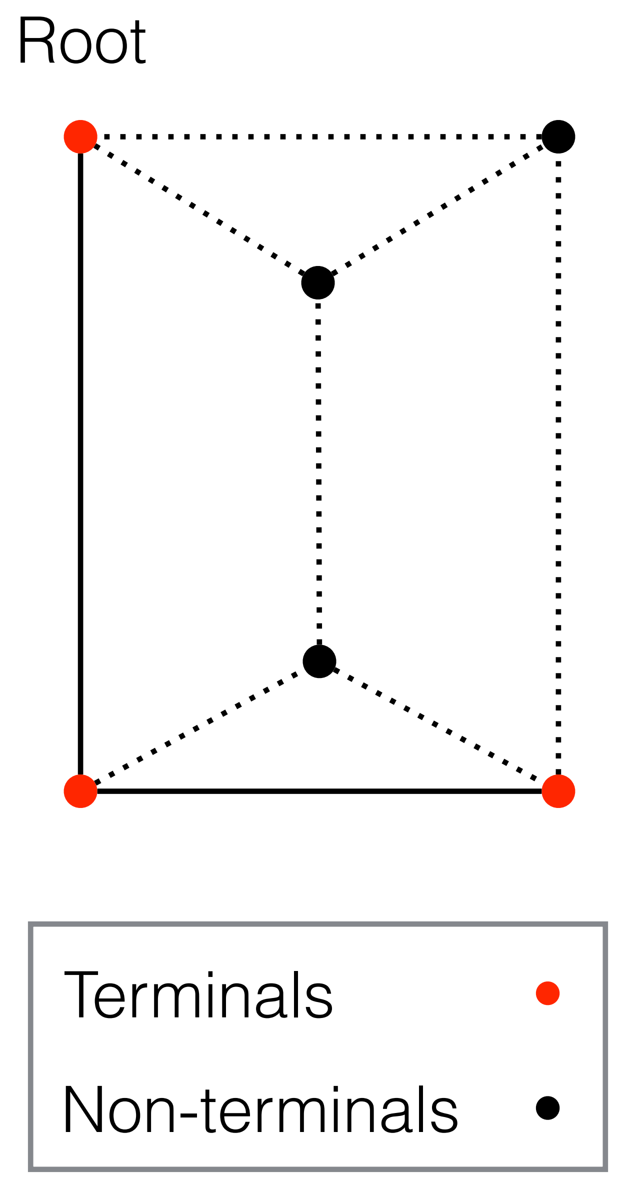
Example



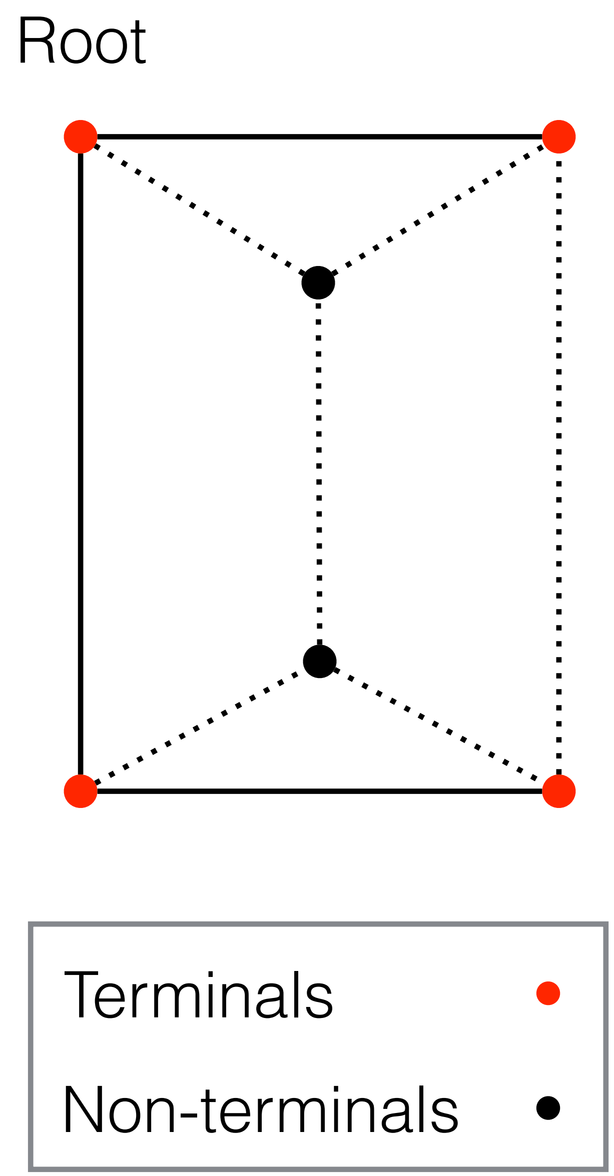
Example



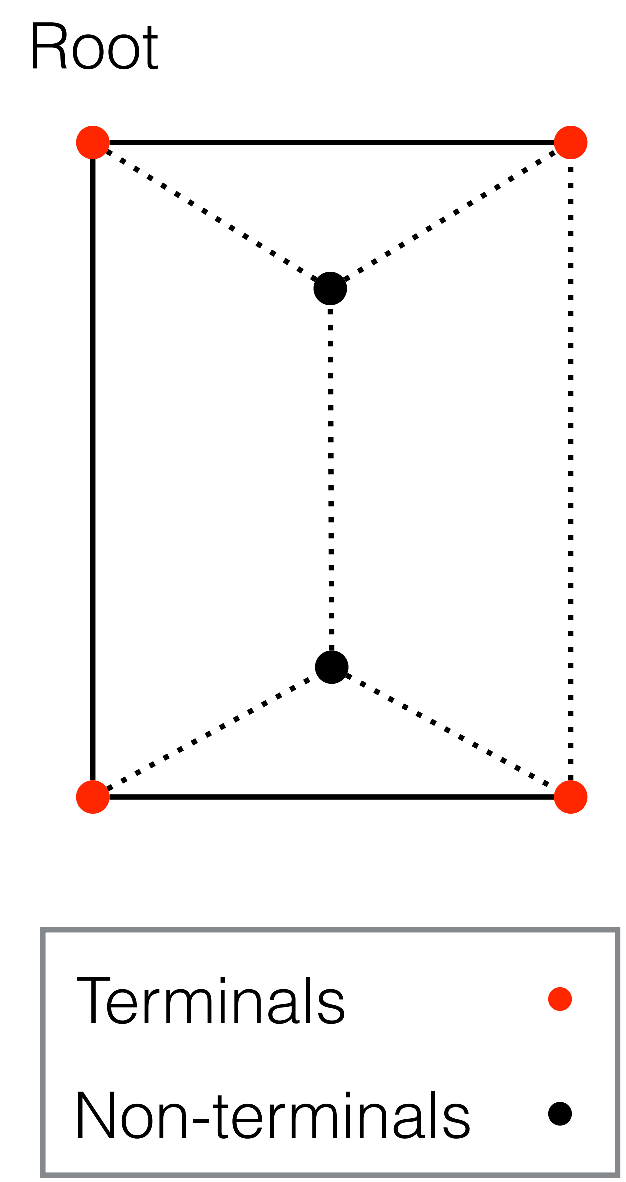
Example



Example



Example



Competitive Ratio: $\displaystyle \max_{\substack{\text{requirements } R \\ \text{sequence } \sigma}} \frac{\text{ALG}(R, \sigma)}{\text{OPT}(R)}$

Problems

Problems

- Steiner forest: "connect terminals s_i and t_i"

Problems

- Steiner forest: “connect terminals s_i and t_i”
- Steiner network: “connect terminals s_i and t_i with R_i edge-disjoint paths, edge-duplication allowed”

Problems

- Steiner forest: “connect terminals s_i and t_i”
- Steiner network: “connect terminals s_i and t_i with R_i edge-disjoint paths, edge-duplication allowed”
- Prize-collecting Steiner forest: “connect terminals s_i and t_i or pay penalty p_i”

Problems

- Steiner forest: “connect terminals s_i and t_i”
- Steiner network: “connect terminals s_i and t_i with R_i edge-disjoint paths, edge-duplication allowed”
- Prize-collecting Steiner forest: “connect terminals s_i and t_i or pay penalty p_i”
- Rent-or-Buy

Problems

- Steiner forest: “connect terminals s_i and t_i”
- Steiner network: “connect terminals s_i and t_i with R_i edge-disjoint paths, edge-duplication allowed”
- Prize-collecting Steiner forest: “connect terminals s_i and t_i or pay penalty p_i”
- Rent-or-Buy
- Connected Facility Location

Results

Problem	Approximation Ratio
Steiner tree Steiner forest	1.39 [Byrka-Grandoni-Rothvoss-Sanita 10] 2 [Agrawal-Klein-Ravi 91; Goemans-Williamson 92]
PC Steiner tree PC Steiner forest	2 [Goemans-Williamson 95] 2.54 [Hajiaghayi-Jain 06]
Steiner network	2 [Jain 98]
Rent-or-buy	3.19 [Grandoni-Rothvoss 11]
Connected facility location	3.19 [Grandoni-Rothvoss 11]

Results

Problem	Approx. Ratio	Previous Competitive Ratios
Steiner tree Steiner forest	1.39 2	O(log k), Ω(log k) [Imase-Waxman 91] O(log k) [Berman-Coulston 97]
PC Steiner tree PC Steiner forest	2 2.54	O(log k) [Qian-Williamson 11]
Steiner network	2	O(log n) randomized [folklore]
Rent-or-buy	3,19	O($\log^2$ k) deterministic, O(log k) randomized [Awerbuch-Azar-Bartal 96]
Connected facility location	3,19	O($\log^2$ k) randomized [San Felice-Williamson-Lee 14]

k = # of terminals; n = # of vertices

Results

Problem	Approx. Ratio	Prev. Comp. Ratios	Our Comp. Ratios
Steiner tree **Steiner forest**	1.39 2	O(log k), Ω(log k) O(log k)	O(log k) O(log k) (simpler proof)
PC Steiner tree **PC Steiner forest**	2 2.54	O(log k)	O(log k) (simpler proof)
Steiner network	2	O(log n) randomized	O(log k) deterministic
Rent-or-buy	3,19	O($\log^2$ k) deterministic O(log k) randomized	O(log k) deterministic
Connected facility location	3,19	O($\log^2$ k) randomized	O(log k) deterministic

k = # of terminals; n = # of vertices

Results

Problem	Approx. Ratio	Prev. Comp. Ratios	Our Comp. Ratios
Steiner tree Steiner forest	1.39 2	O(log k), **Ω**(log k) O(log k)	O(log k) O(log k) (simpler proof)
PC Steiner tree PC Steiner forest	2 2.54	O(log k)	O(log k) (simpler proof)
Steiner network	2	O(log n) randomized	O(log k) deterministic
Rent-or-buy	3,19	O($\log^2$ k) deterministic O(log k) randomized	O(log k) deterministic
Connected facility location	3,19	O($\log^2$ k) randomized*	O(log k) deterministic

k = # of terminals; n = # of vertices

Results

Problem	Approx. Ratio	Prev. Comp. Ratios	Our Comp. Ratios
Steiner tree **Steiner forest**	1.39 2	O(log k), **Ω**(log k) O(log k)	O(log k) O(log k) (simpler proof)
PC Steiner tree **PC Steiner forest**	2 2.54	O(log k)	O(log k) (simpler proof)
Steiner network	2	O(log n) randomized	O(log k) deterministic
Rent-or-buy	3,19	$O(\log^2 k)$ deterministic O(log k) randomized	O(log k) deterministic
Connected facility location	3,19	$O(\log^2 k)$ randomized*	O(log k) deterministic

k = # of terminals; n = # of vertices

Results

Problem	Approx. Ratio	Prev. Comp. Ratios	Our Comp. Ratios
Steiner tree Steiner forest	1.39 2	$O(\log k)$, $\boldsymbol{\Omega}(\log k)$ $O(\log k)$	$O(\log k)$ $O(\log k)$ (simpler proof)
PC Steiner tree PC Steiner forest	2 2.54	$O(\log k)$	$O(\log k)$ (simpler proof)
Steiner network	2	$O(\log n)$ randomized	$O(\log k)$ deterministic
Rent-or-buy	3,19	$O(\log^2 k)$ deterministic $O(\log k)$ randomized	$O(\log k)$ deterministic
Connected facility location	3,19	$O(\log^2 k)$ randomized*	$O(\log k)$ deterministic

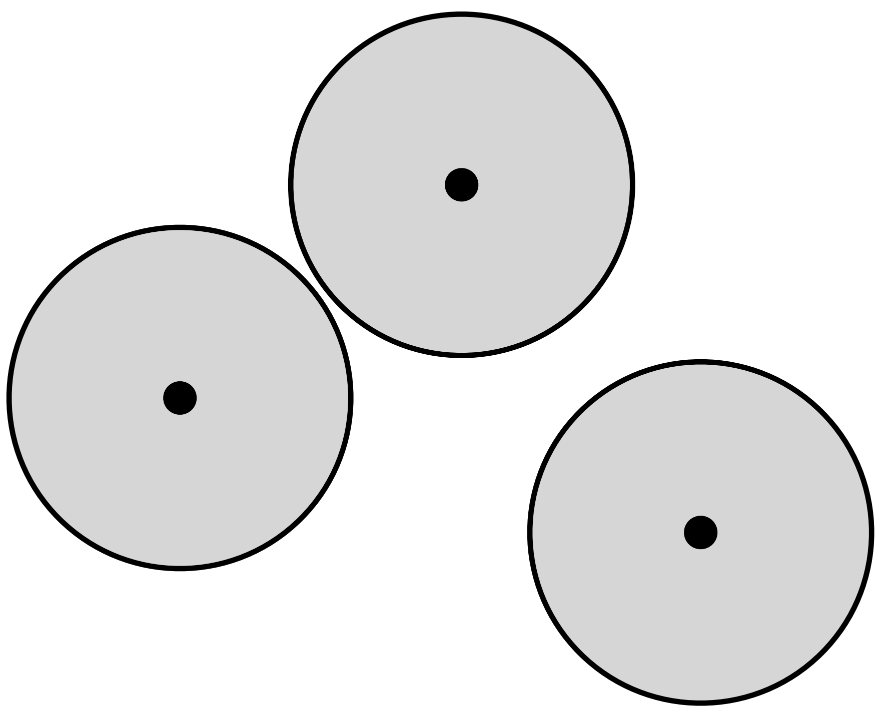
(*Independently improved to $O(\log k)$ randomized)

k = # of terminals; n = # of vertices

Previous Approaches

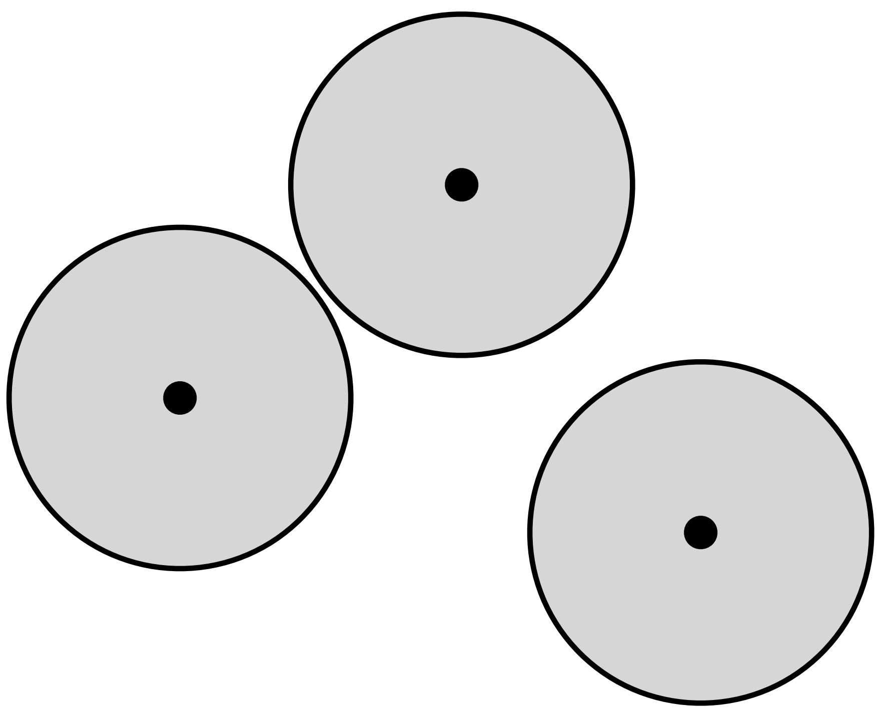
Previous Approaches

Greedy / Primal-Dual

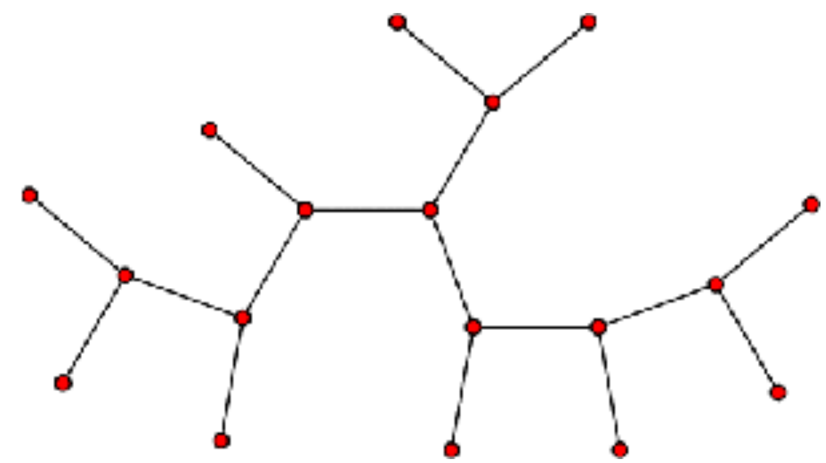


Previous Approaches

Greedy / Primal-Dual

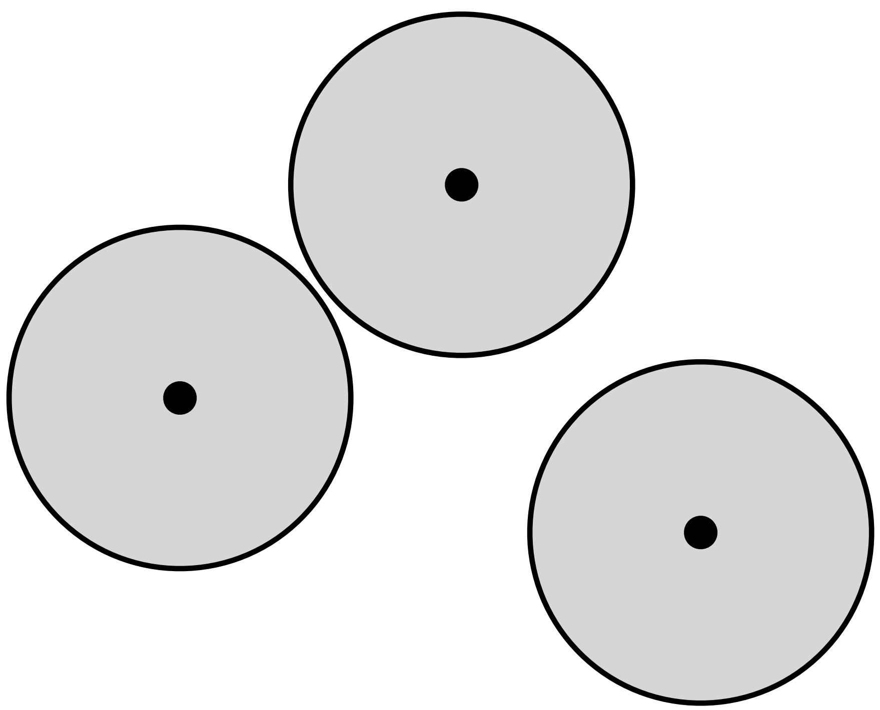


Tree Embeddings



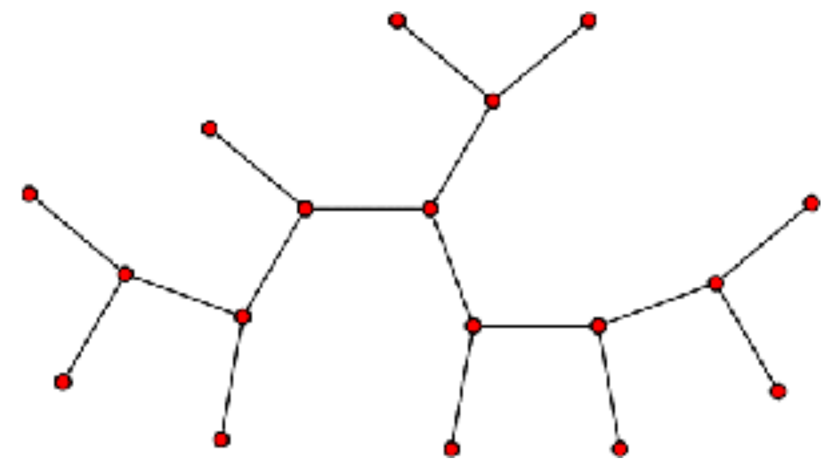
Previous Approaches

Greedy / Primal-Dual



- Deterministic algorithms

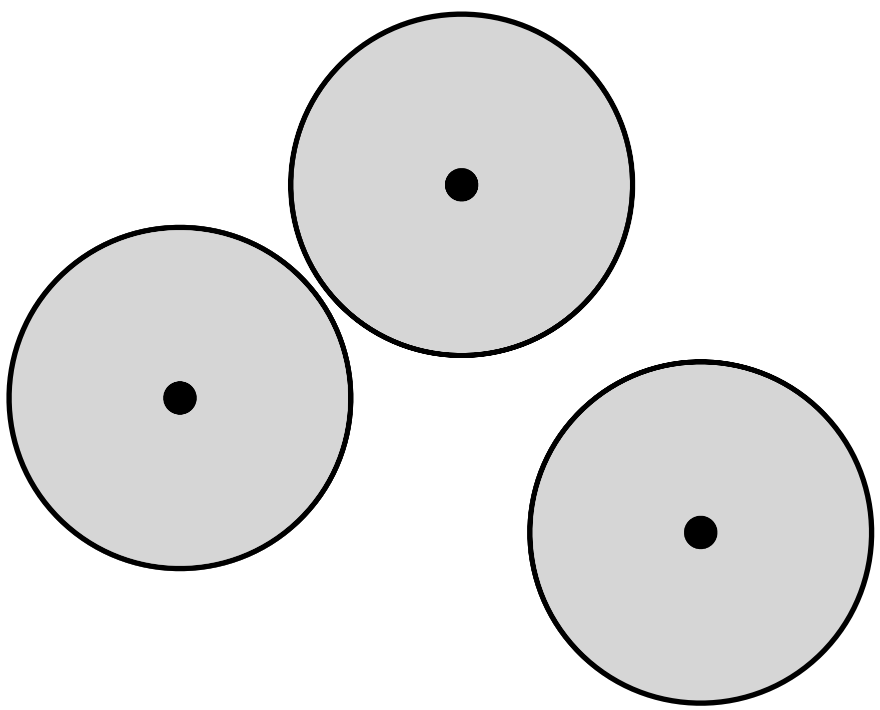
Tree Embeddings



- Randomized algorithms

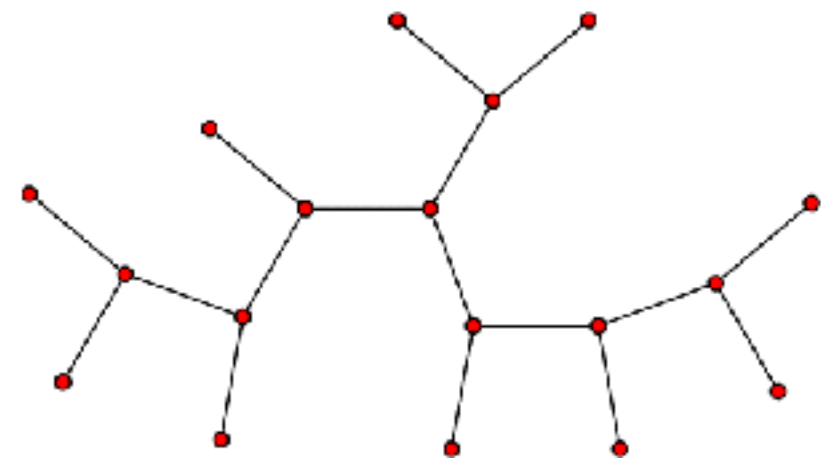
Previous Approaches

Greedy / Primal-Dual



- Deterministic algorithms
- Intricate dual construction

Tree Embeddings



- Randomized algorithms
- Simple analysis

Our Approach

- Greedy algorithms but analyze using tree embeddings
- Deterministic algorithms and simple analyses
- Unified approach to online network design

Our Approach

- Greedy algorithms but analyze using tree embeddings
- Deterministic algorithms and simple analyses
- Unified approach to online network design

Takeaway: Tree embeddings and metric decompositions are useful for designing and analyzing greedy algorithms

Outline

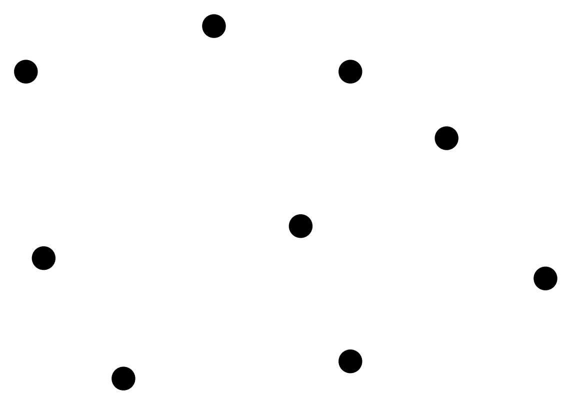
1. Overview of Analysis Framework
2. Warm-Up: Steiner Tree
3. Rent-or-Buy
4. Steiner Network

Metric Problems

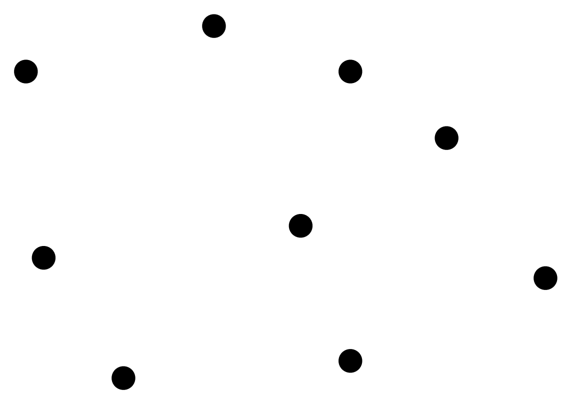
- Up to constants, can assume input graph G is complete and edge costs form a metric
- Henceforth, input is a metric

Tree Embeddings

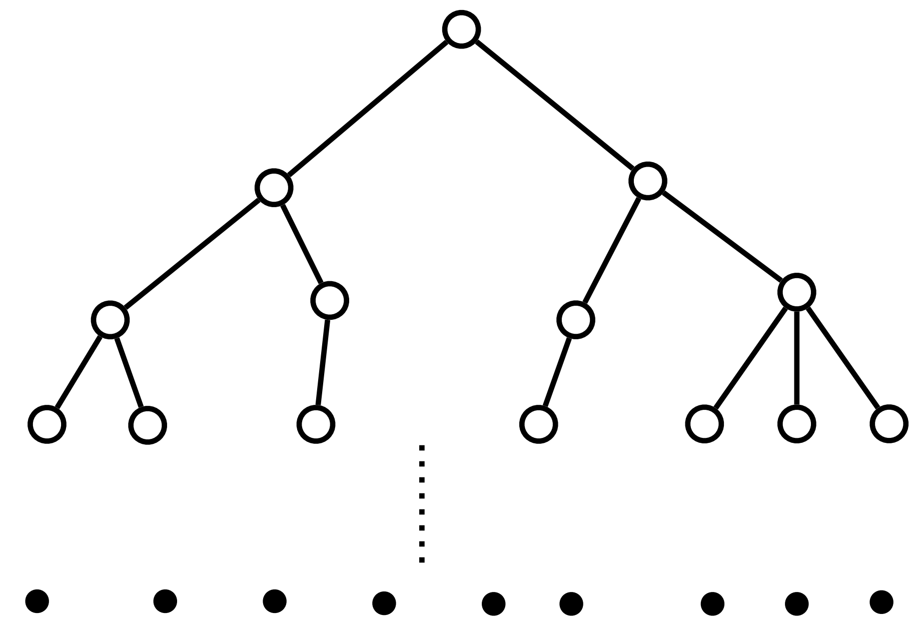
(V, d)



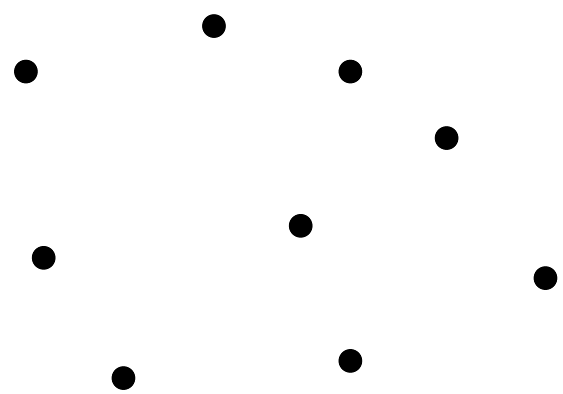
(V, d)



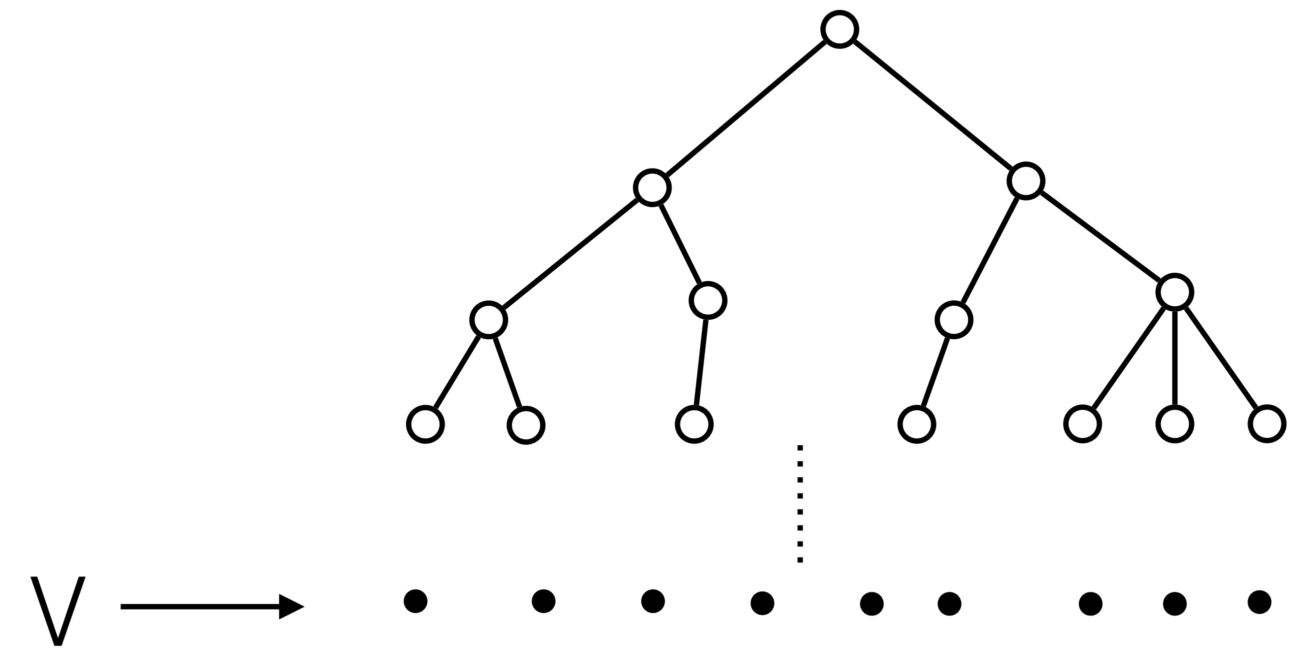
Embedding (V’, T)



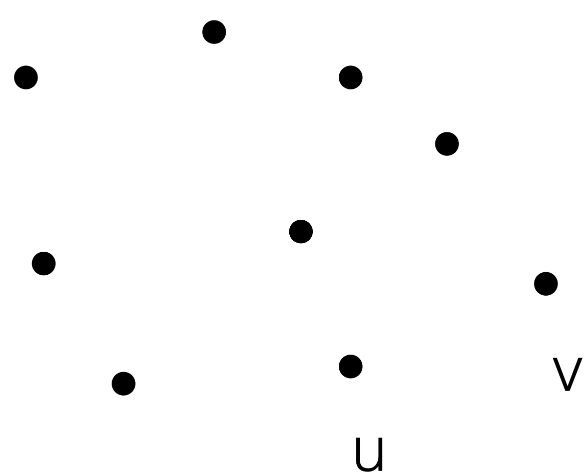
(V, d)



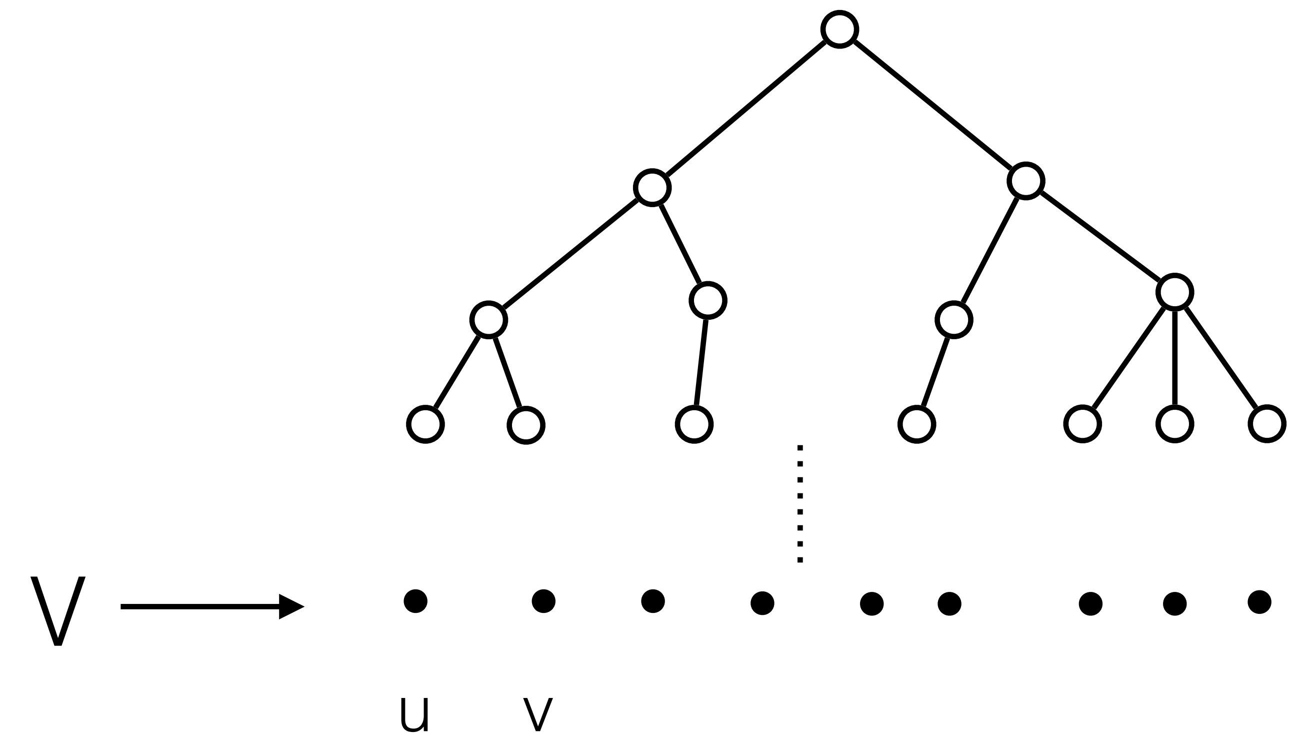
Embedding (V’, T)



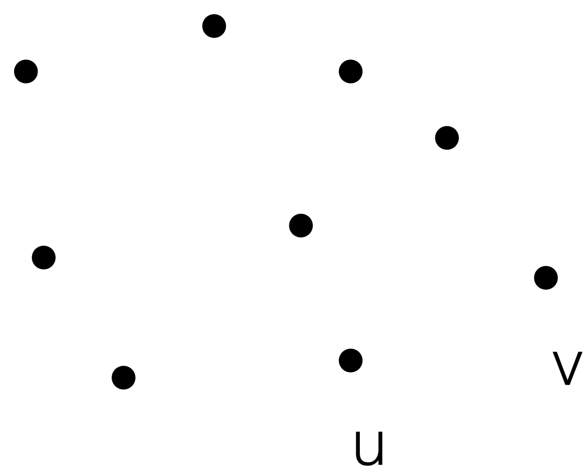
(V, d)



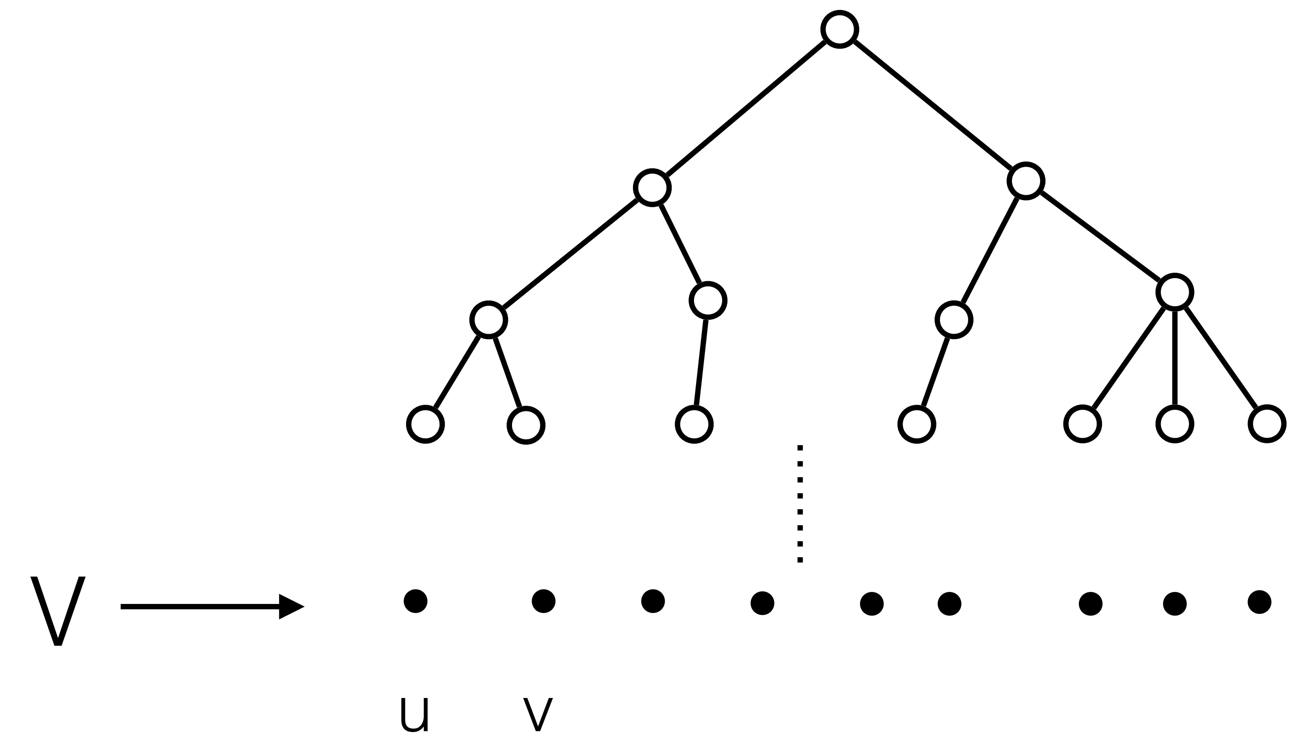
Embedding (V’, T)



(V, d)



Embedding (V', T)

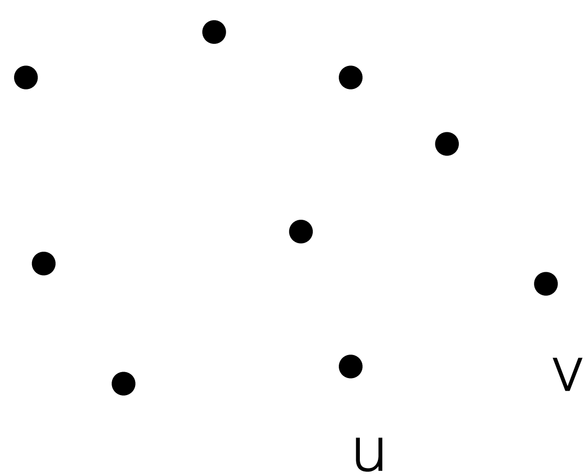


Theorem [Fakcharoenphol-Rao-Talwar 04]:
There exists a randomized embedding into HSTs satisfying:

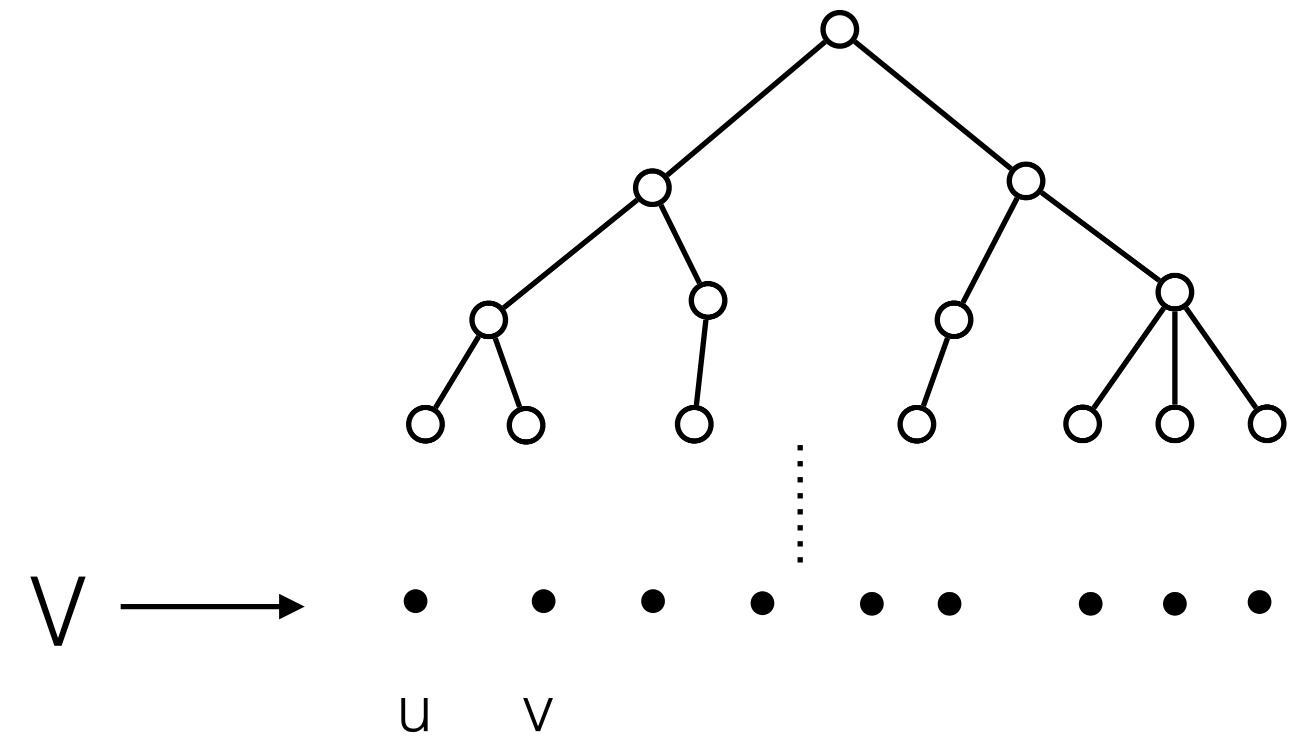
For all u,v in V

$$T(u,v) \geq d(u,v)$$
$$E[T(u,v)] \leq O(\log n)\, d(u,v)$$

(V, d)



Embedding (V', T)



Theorem [Fakcharoenphol-Rao-Talwar 04]:
There exists a randomized embedding into HSTs satisfying:

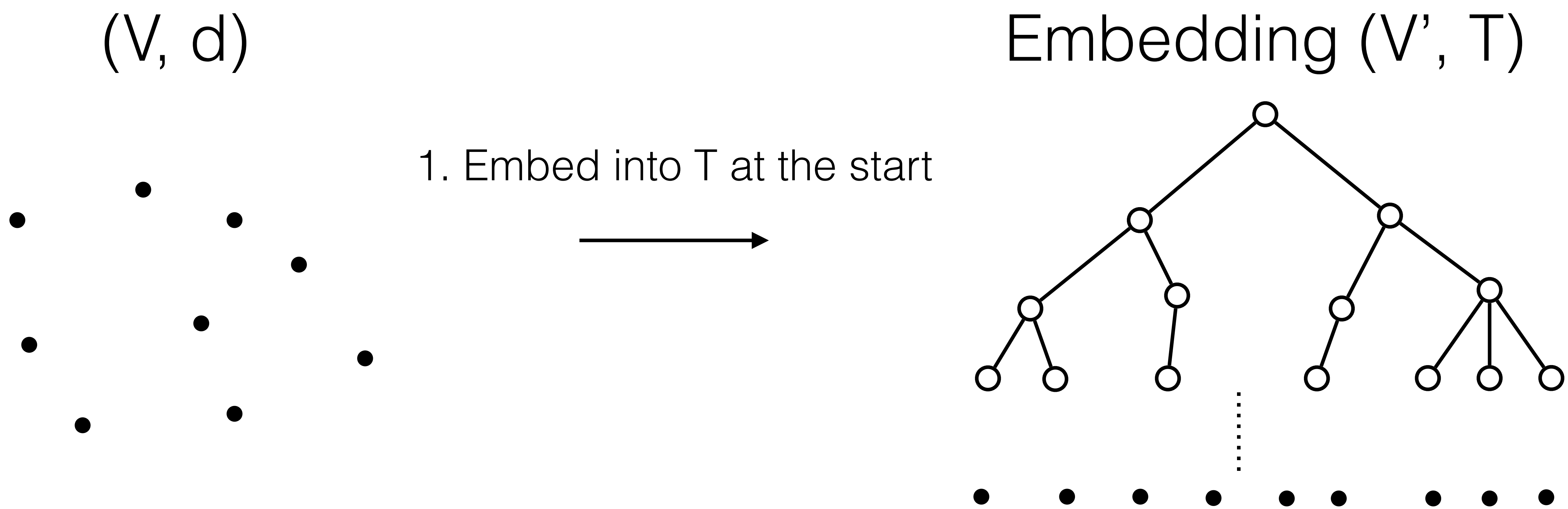
For all u,v in V $\quad T(u,v) \geq d(u,v)$

$$E[T(u,v)] \leq O(\log n)\, d(u,v)$$

Corollary For many network design problems,

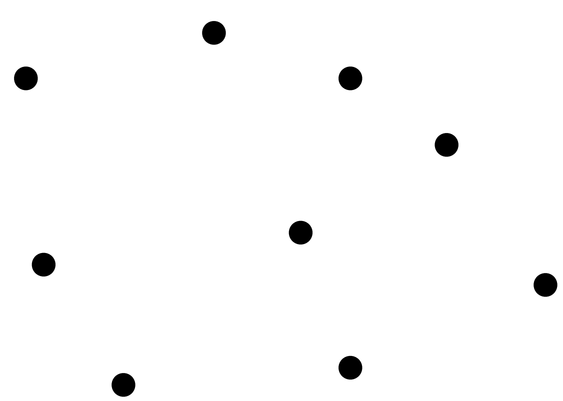
$$OPT \leq E[OPT(T)] \leq O(\log n)\, OPT$$

Previous Online Application



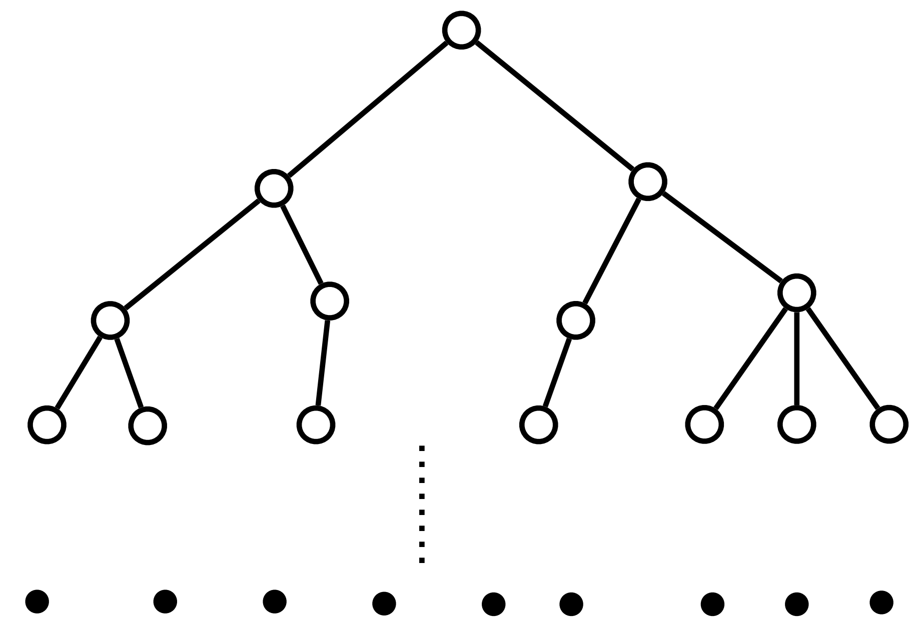
Previous Online Application

(V, d)



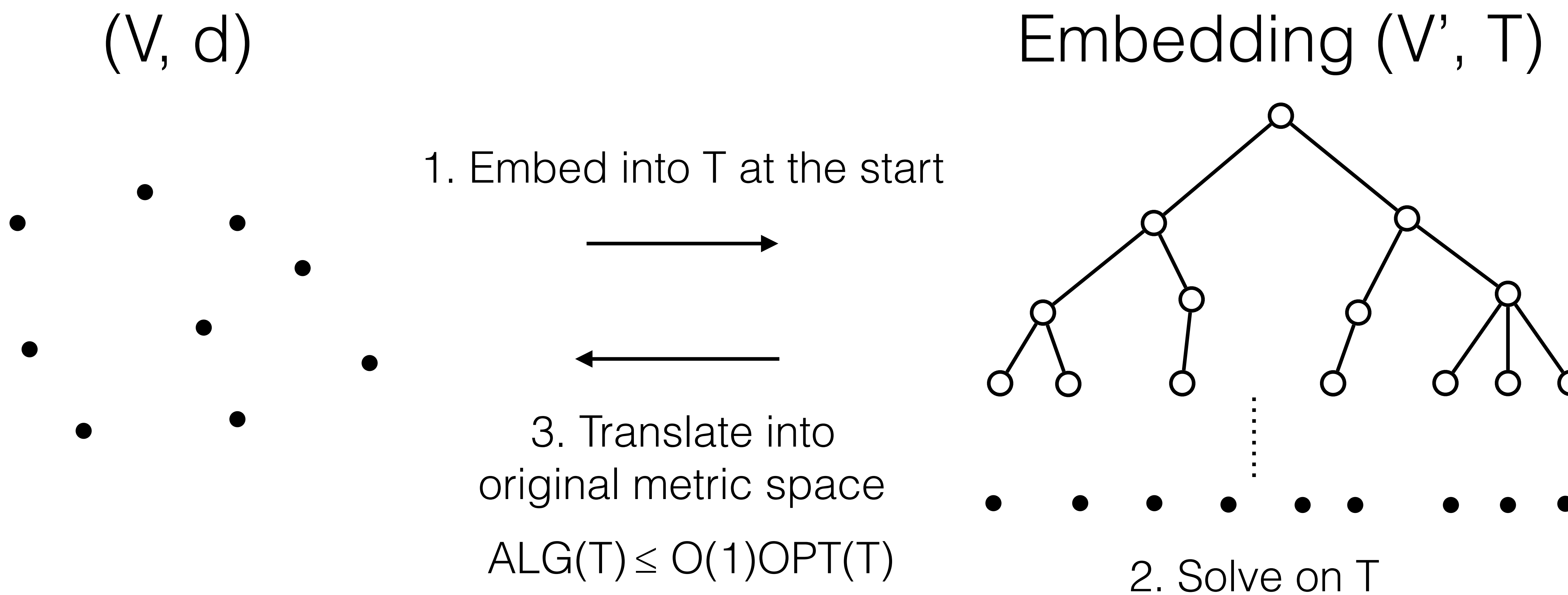
1. Embed into T at the start

Embedding (V', T)



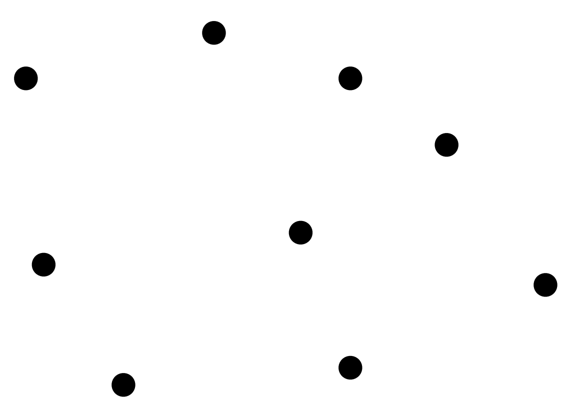
2. Solve on T

Previous Online Application

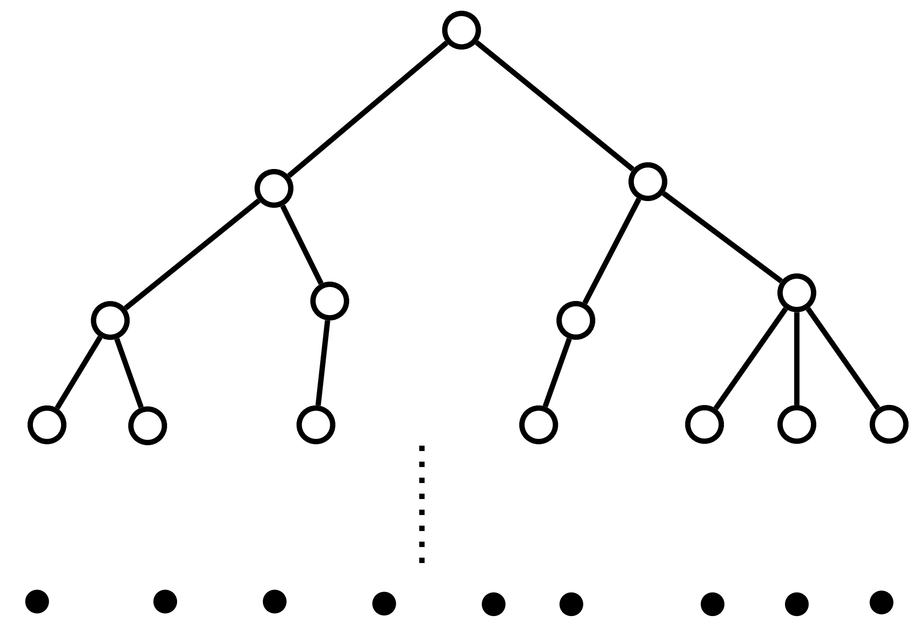


Previous Online Application

(V, d)



Embedding (V', T)



1. Embed into T at the start

2. Solve on T

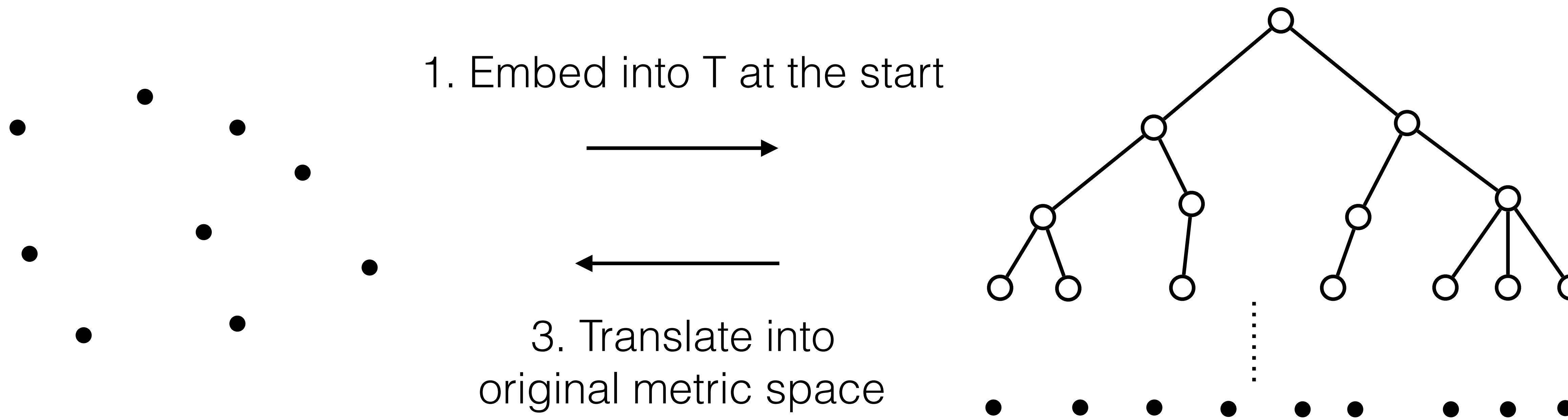
3. Translate into original metric space

ALG(T) ≤ O(1)OPT(T)

E[ALG(T)] ≤ O(log n) OPT

Previous Online Application

(V, d) Embedding (V', T)



ALG(T) ≤ O(1)OPT(T)

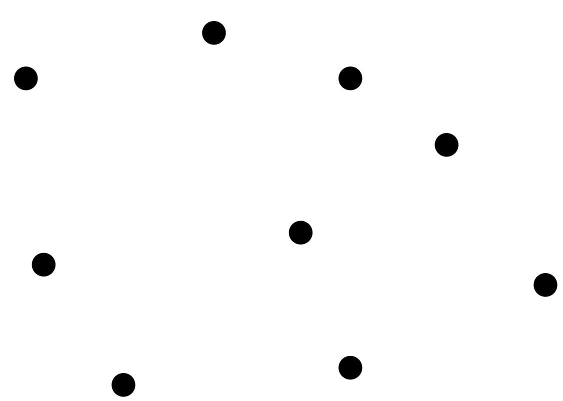
E[ALG(T)] ≤ O(log n) OPT

Drawbacks

- O(log n) competitive ratio even with k ≪ n requests
- Requires randomness

Our Application

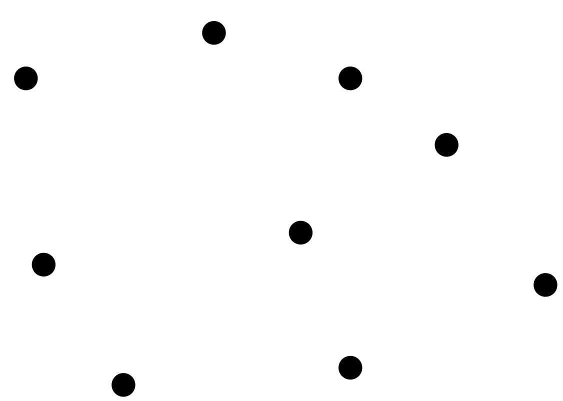
(V, d)



1. Run greedy algorithm ALG

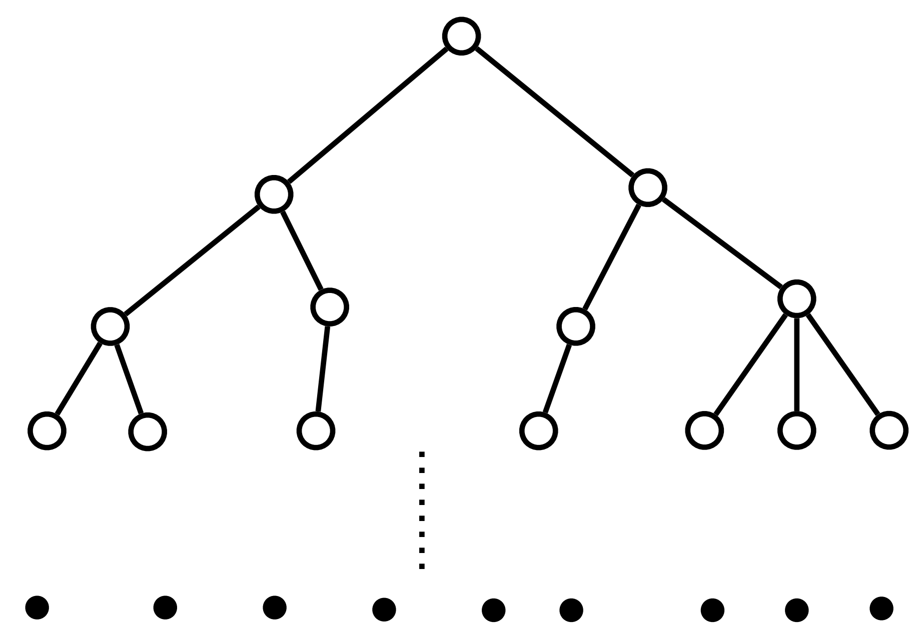
Our Application

(V, d)



1. Run greedy algorithm ALG

Embedding (V’, T)



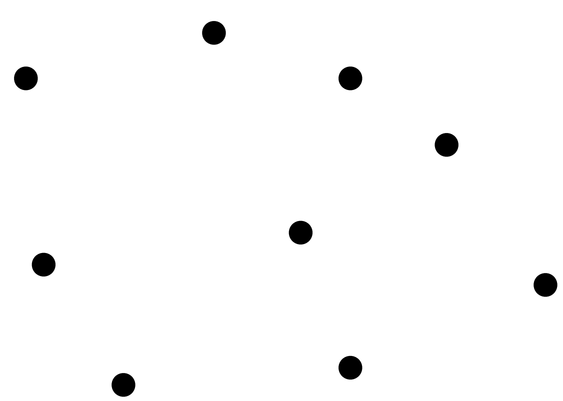
2. Bound ALG against OPT(T)

<u>Main Lemma:</u>

ALG $\leq$ O(1) OPT(T) for *any* HST embedding T

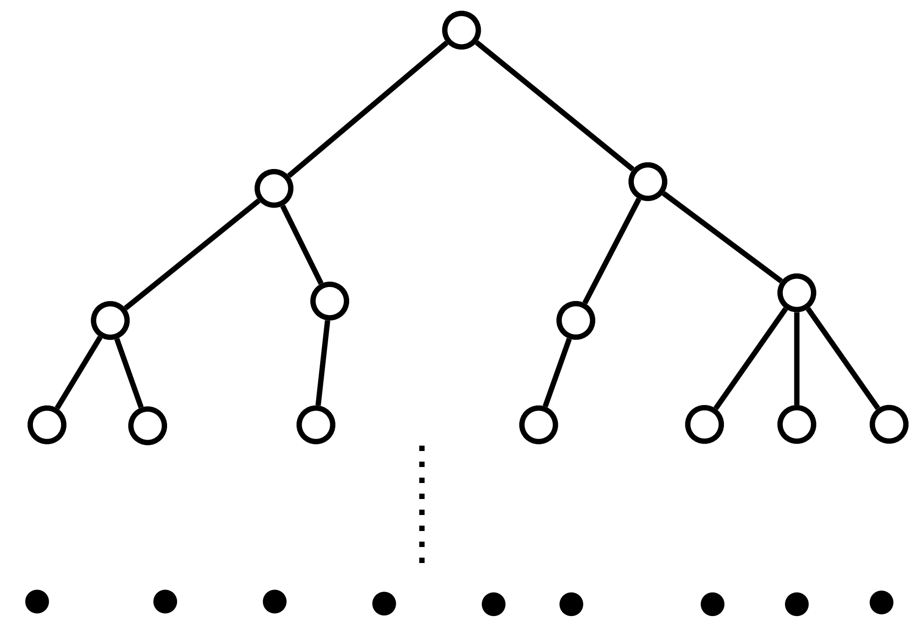
Our Application

(V, d)



1. Run greedy algorithm ALG

Embedding (V’, T)



2. Bound ALG against OPT(T)

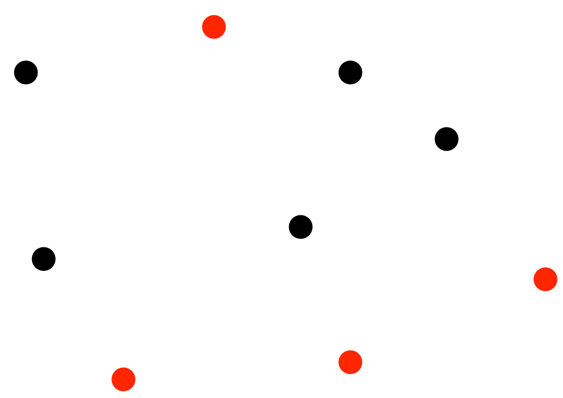
Main Lemma:

ALG $\leq$ O(1) $\min_T$ OPT(T) $\leq$ O(log n) OPT

ALG $\leq$ O(1) OPT(T) for *any* HST embedding T

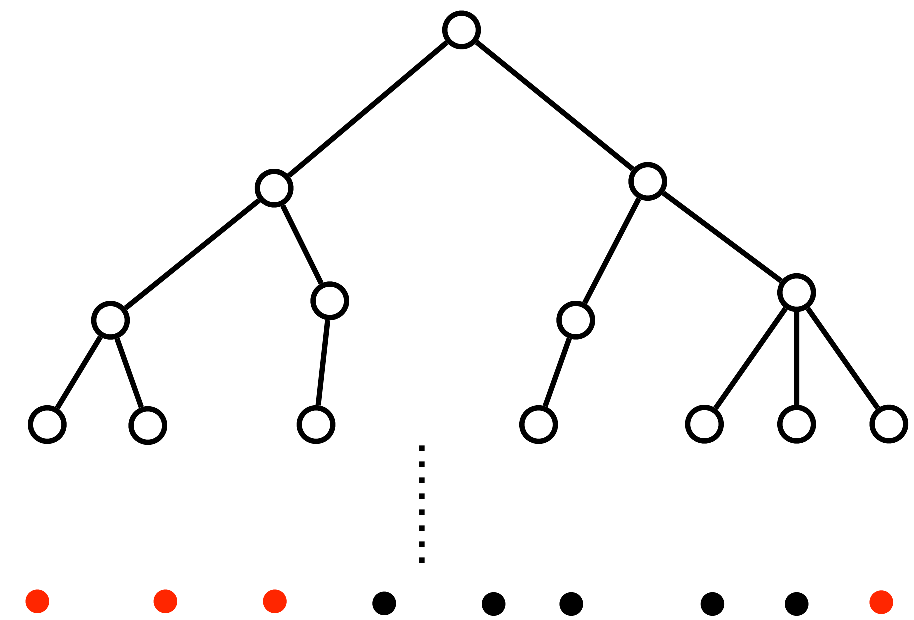
Our Application

(V, d)



1. Run greedy algorithm ALG

Embedding (V’, T)



2. Bound ALG against OPT(T)

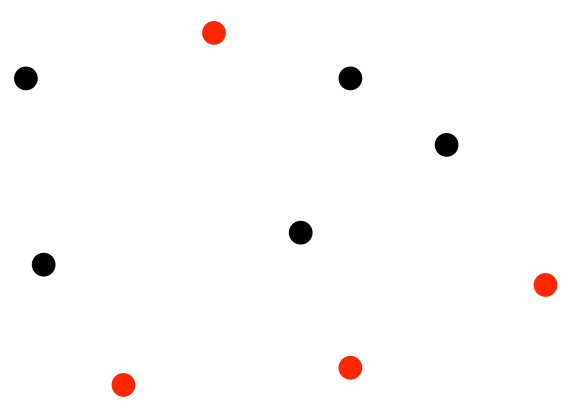
Main Lemma:

ALG ≤ O(1) $\min_T$ OPT(T) ≤ O(log n) OPT

ALG ≤ O(1) OPT(T) for *any* HST embedding T

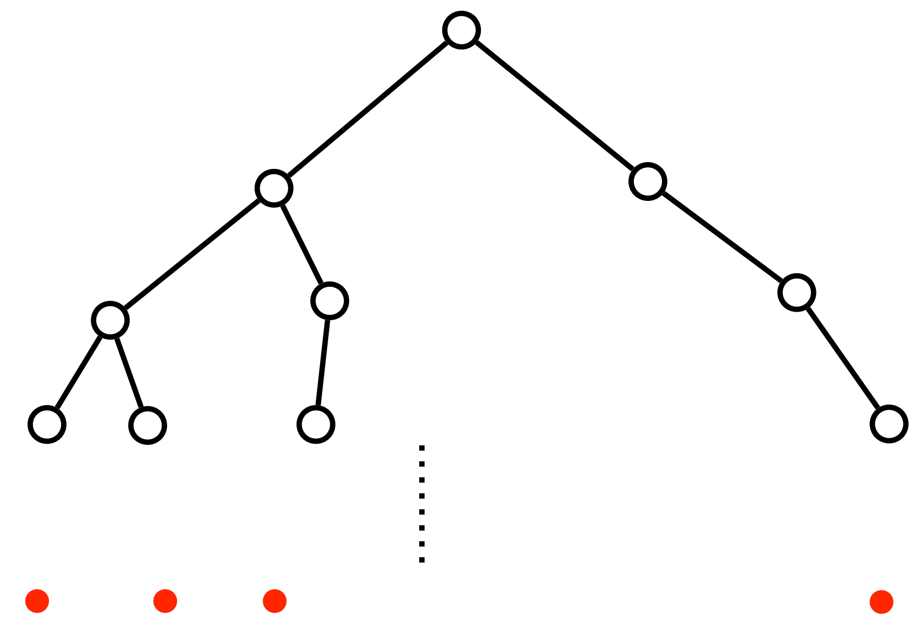
Our Application

(V, d)



1. Run greedy algorithm ALG

Embedding (V’, T)



2. Bound ALG against OPT(T)

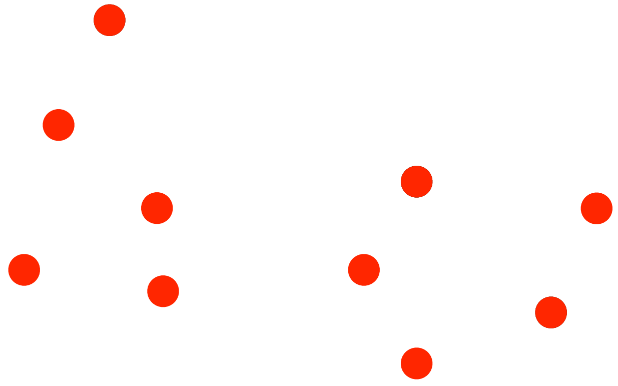
Main Lemma:

ALG $\leq$ O(1) $\min_T$ OPT(T) $\leq$ O(log k) OPT

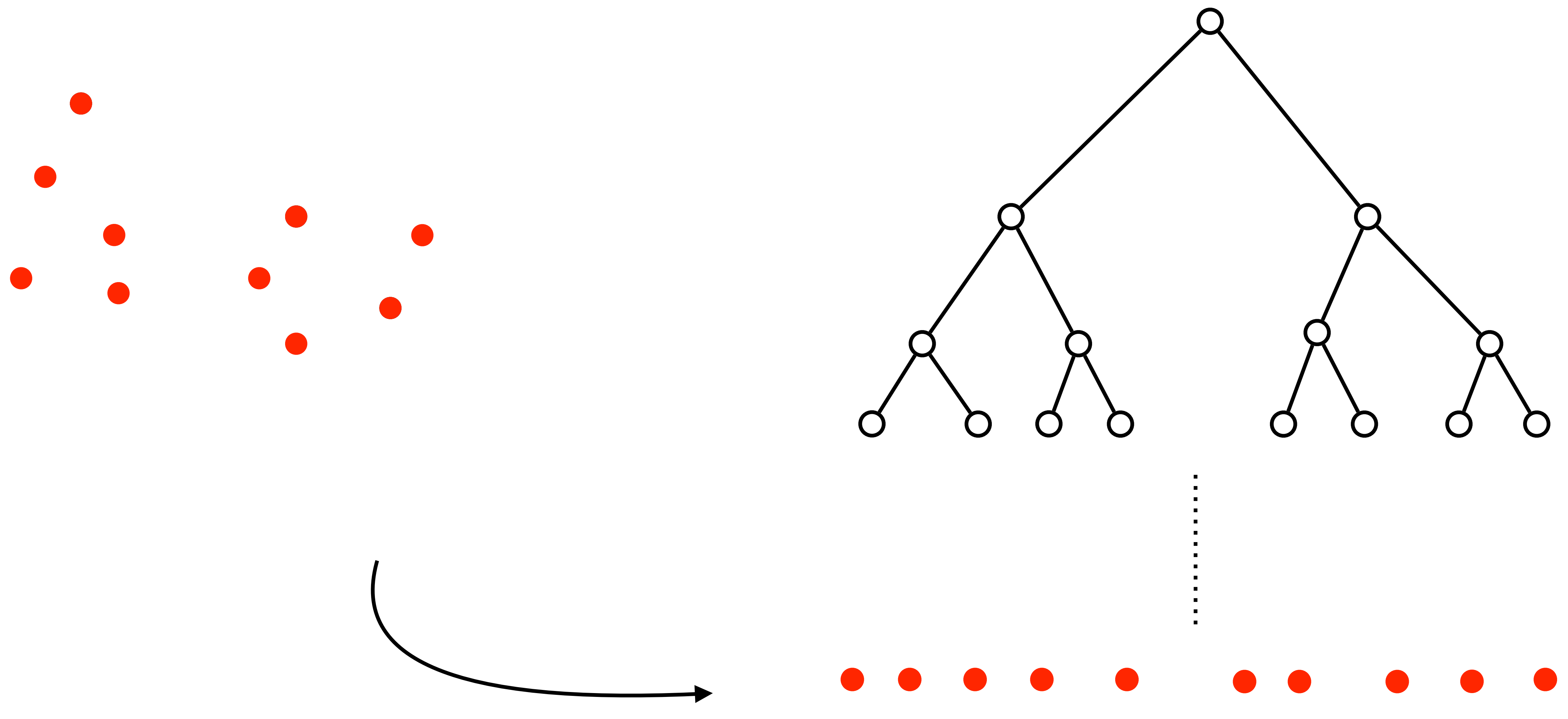
ALG $\leq$ O(1) OPT(T) for *any* HST embedding T of terminals

HST Embeddings

HST Embeddings

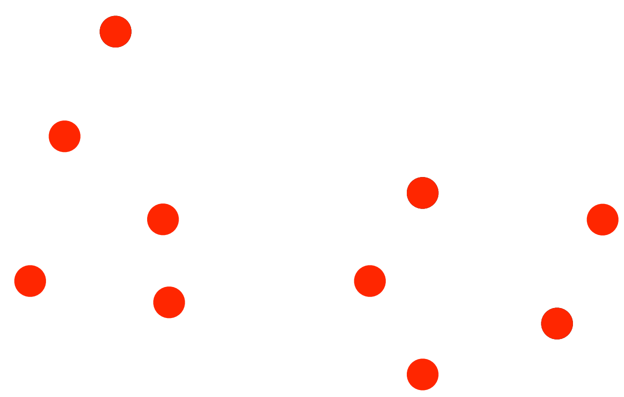


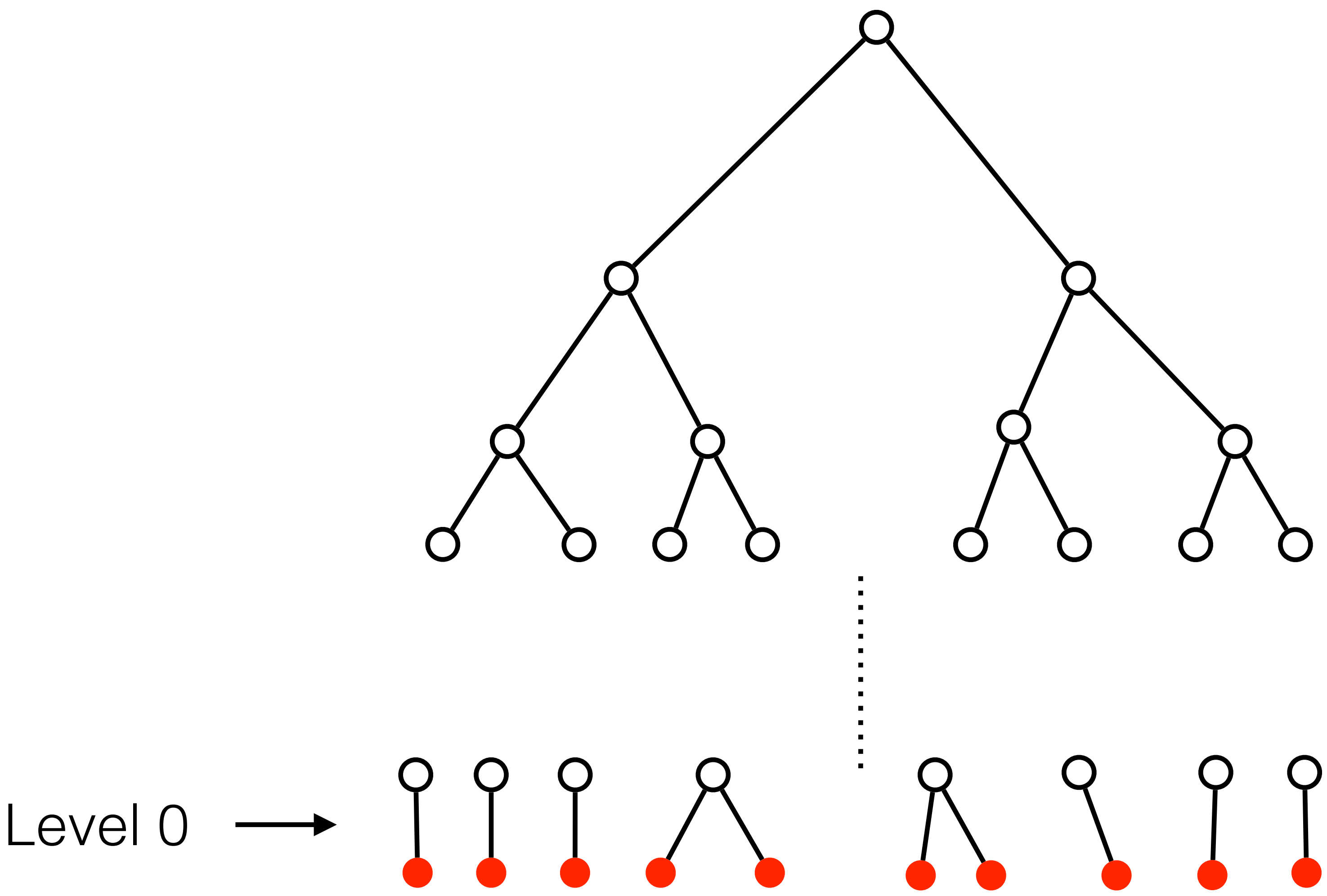
HST Embeddings



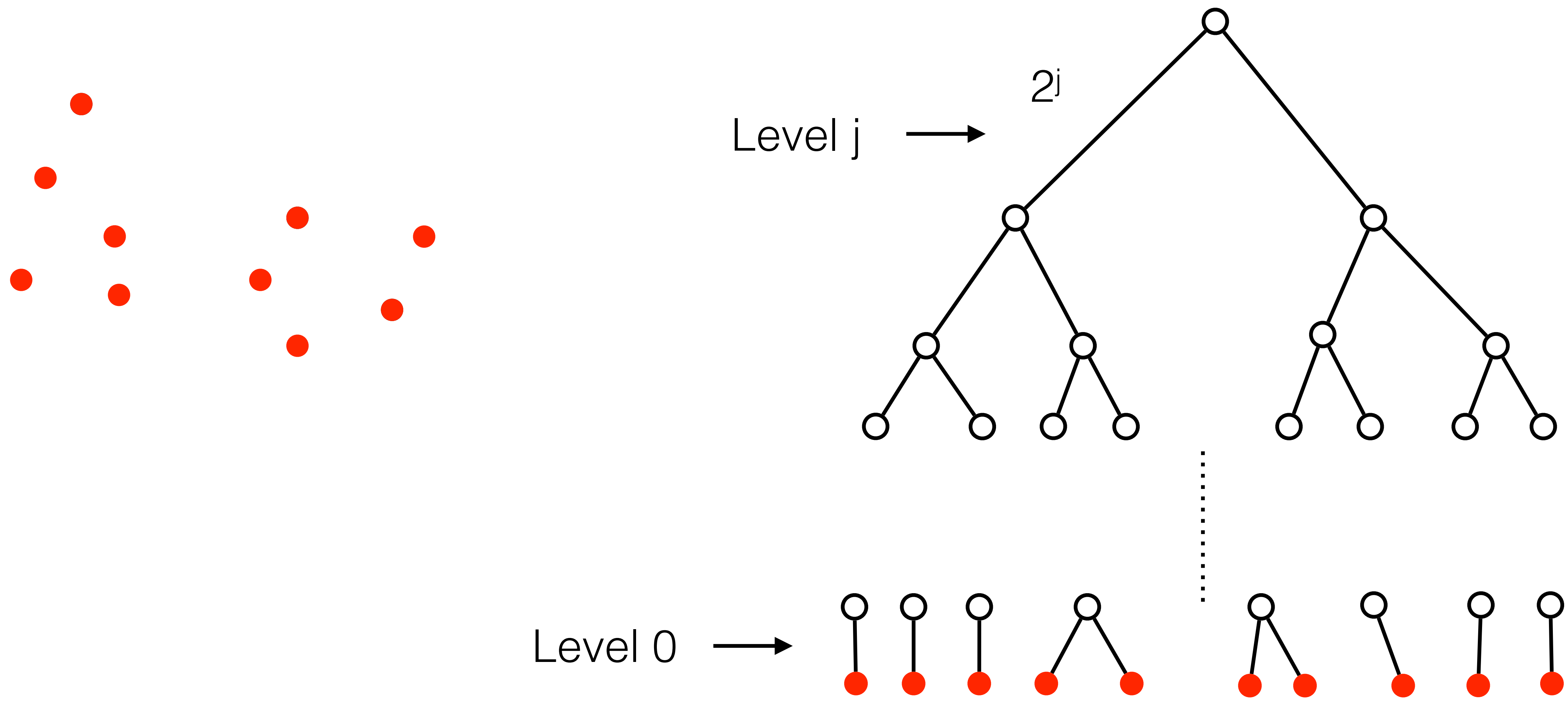
Terminals → leaves of T

HST Embeddings

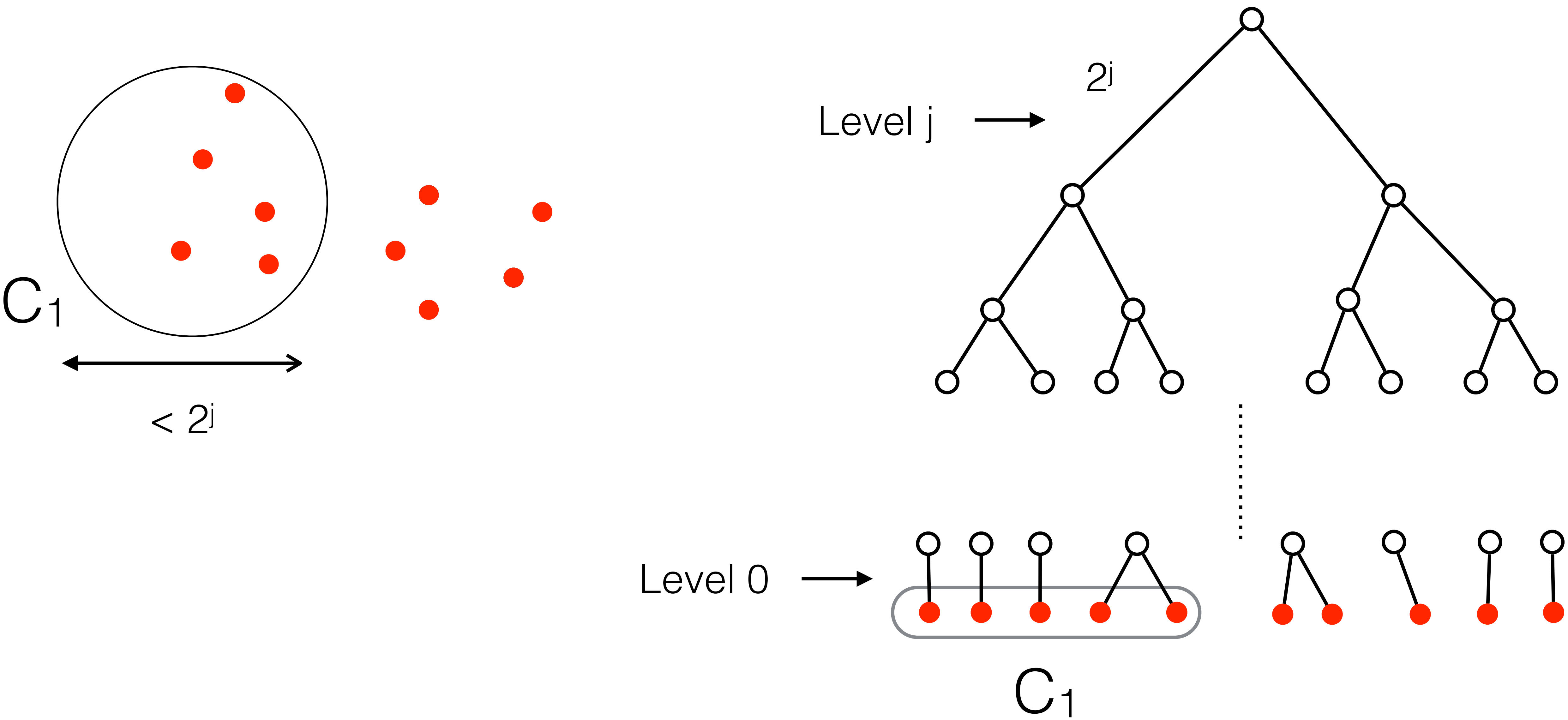




HST Embeddings

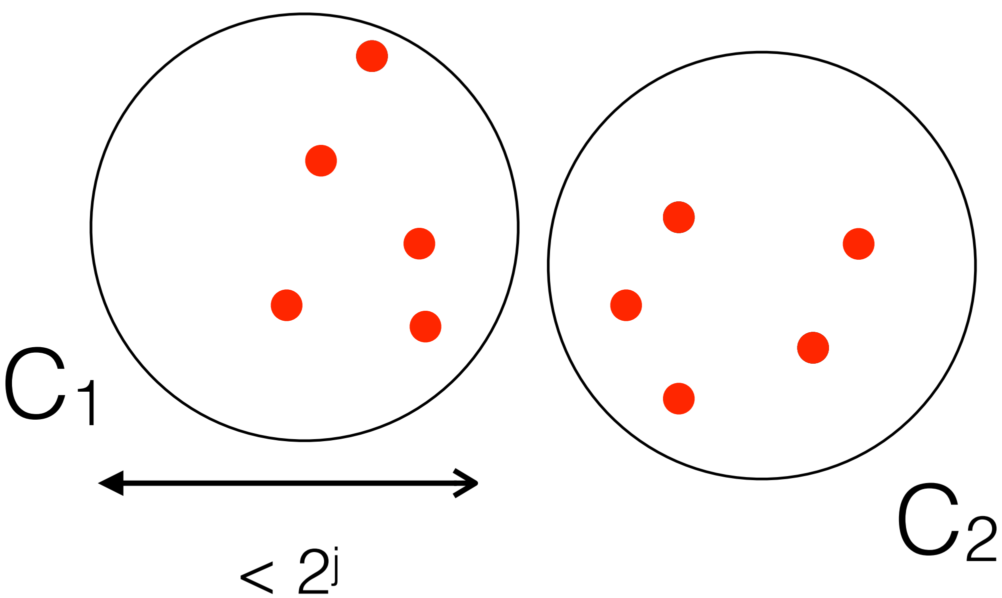


HST Embeddings

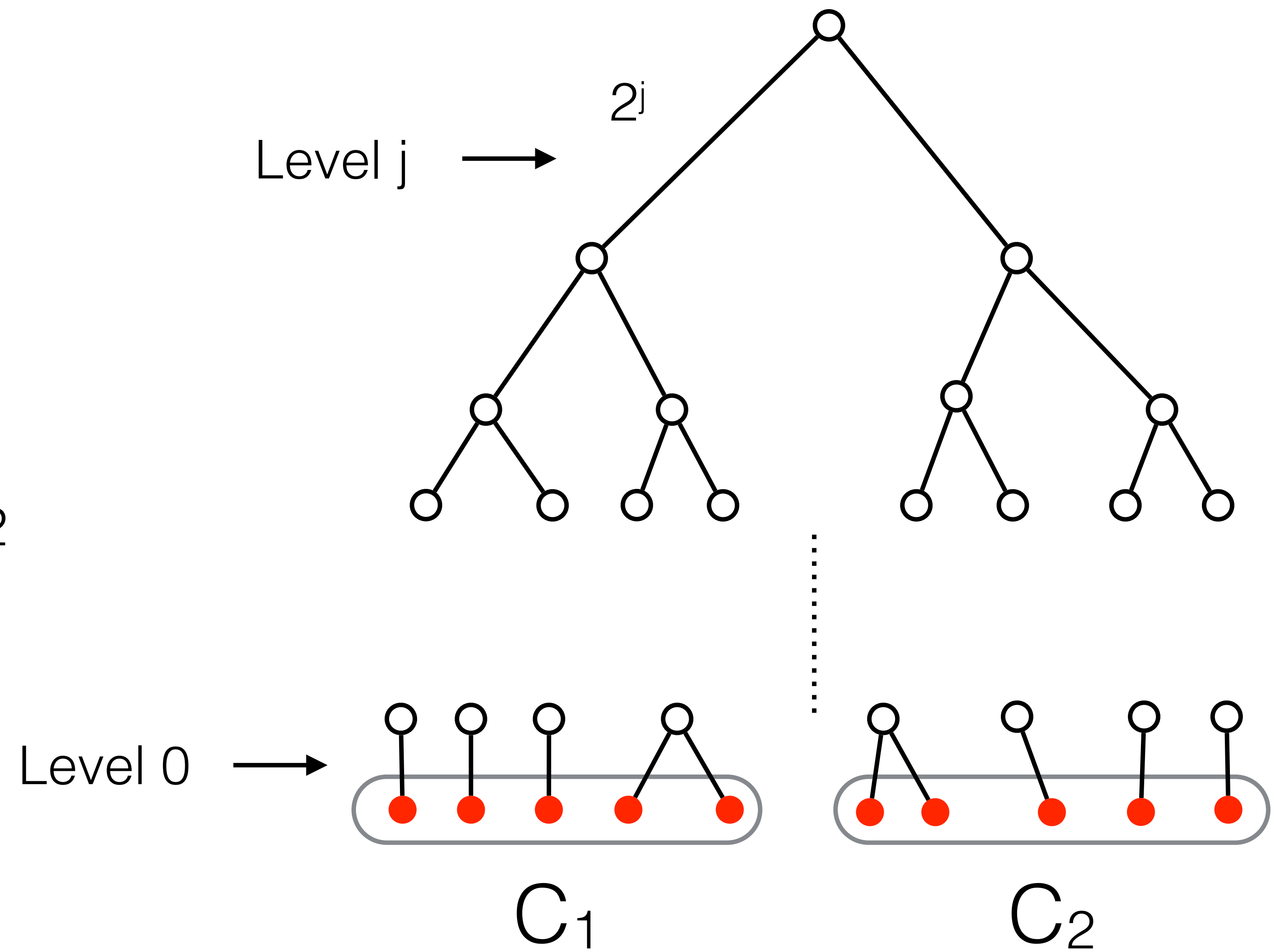


1. Leaves of level-j edge → 2^j-diam subset

HST Embeddings



2^j-diam partition



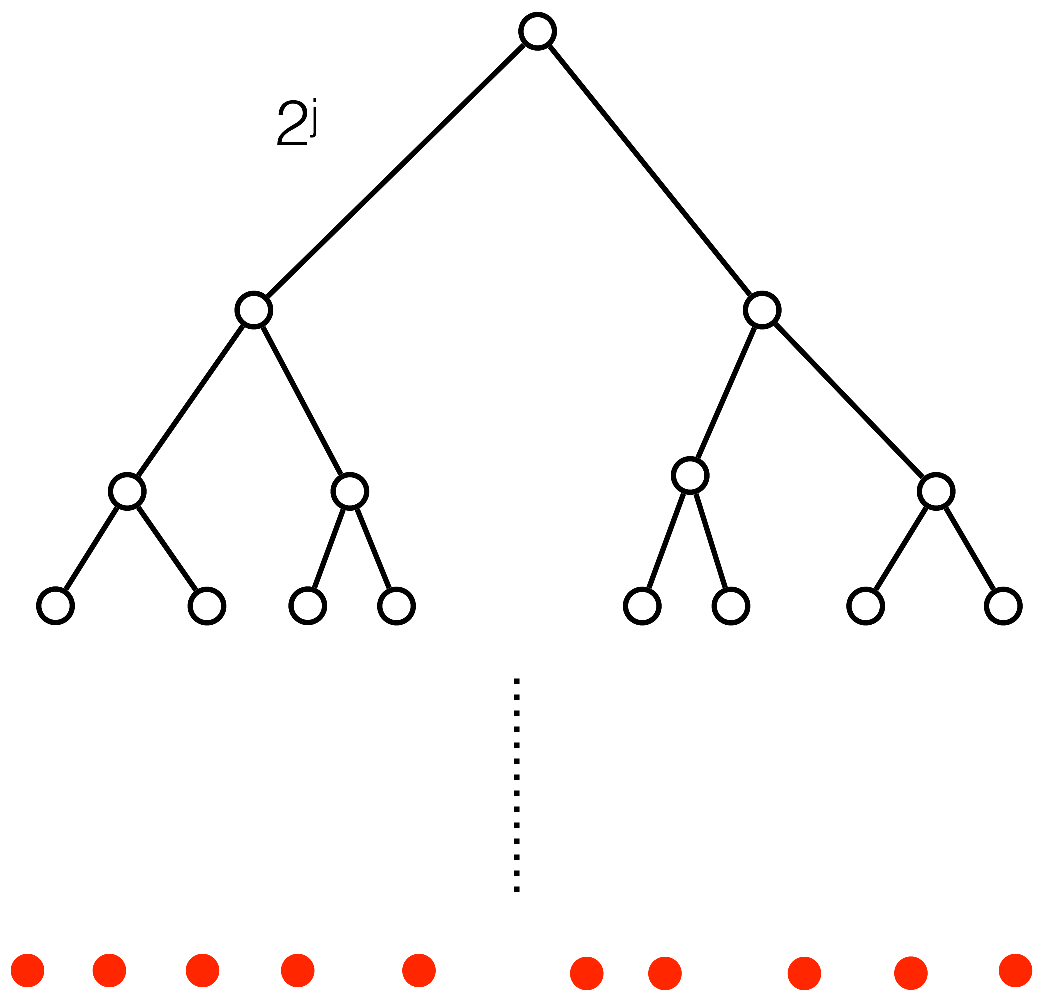
1. Leaves of level-j edge → 2^j-diam subset
2. Level-j edges → 2^j-diam decomposition

Goal: ALG $\leq$ O(1) OPT(T) for any HST embedding T

Proof Strategy:

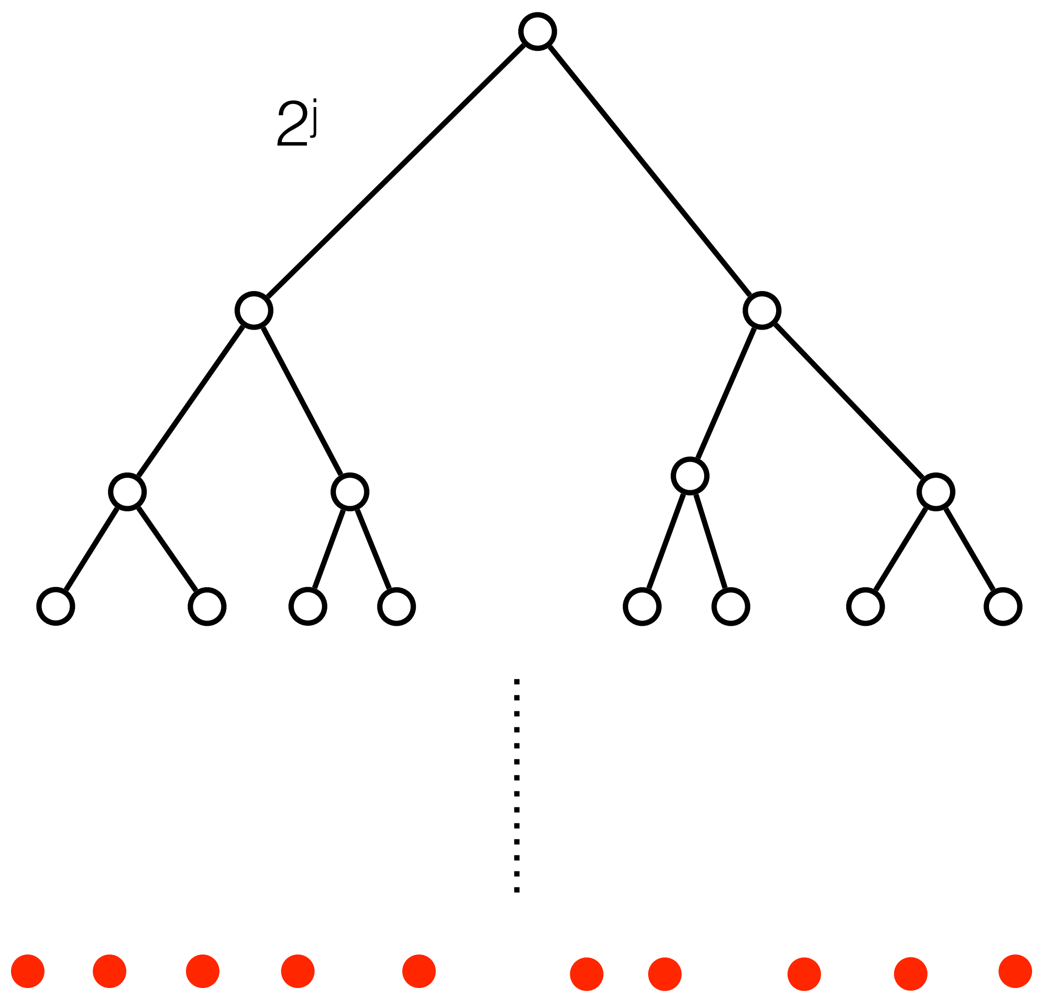
Goal: ALG $\leq$ O(1) OPT(T) for any HST embedding T

Proof Strategy:



Goal: ALG ≤ O(1) OPT(T) for any HST embedding T

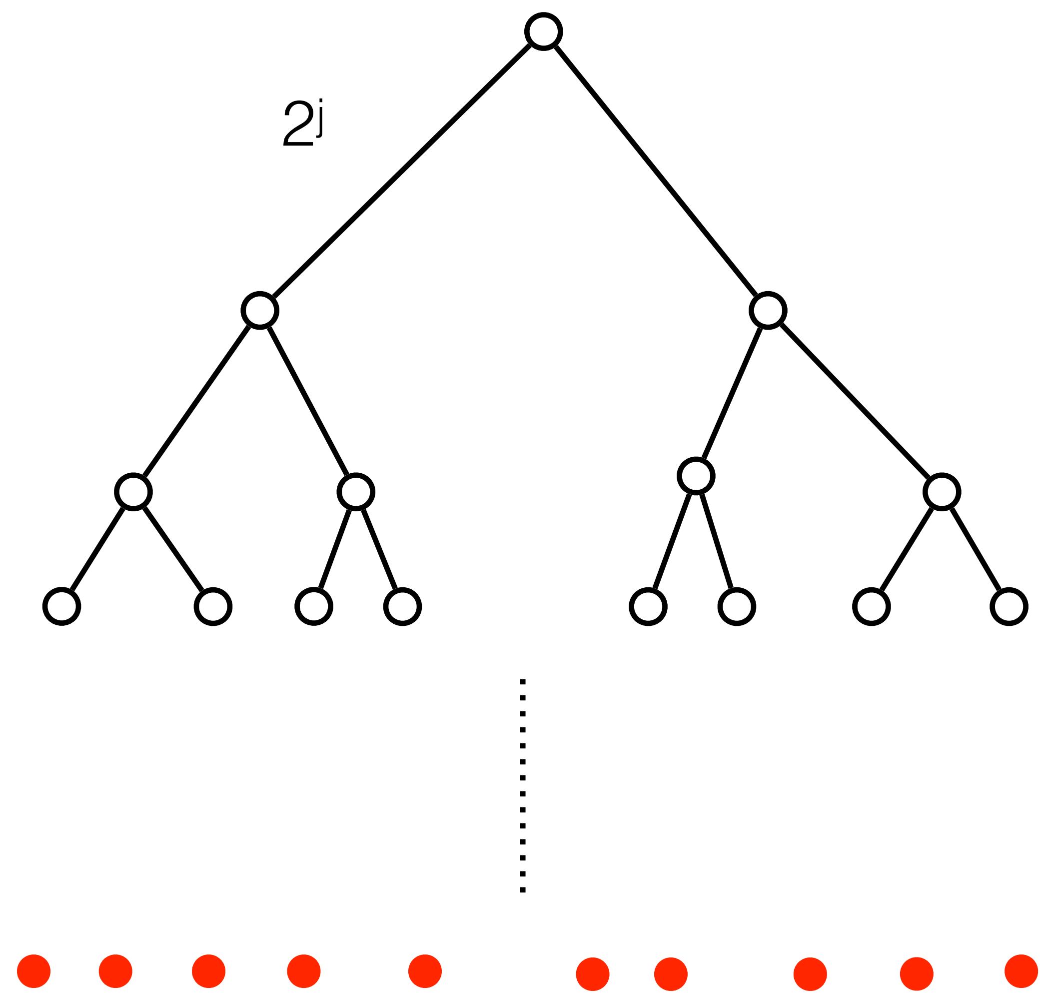
Proof Strategy:



1. Decompose OPT(T) into contributions from tree edges

Goal: ALG $\leq$ O(1) OPT(T) for any HST embedding T

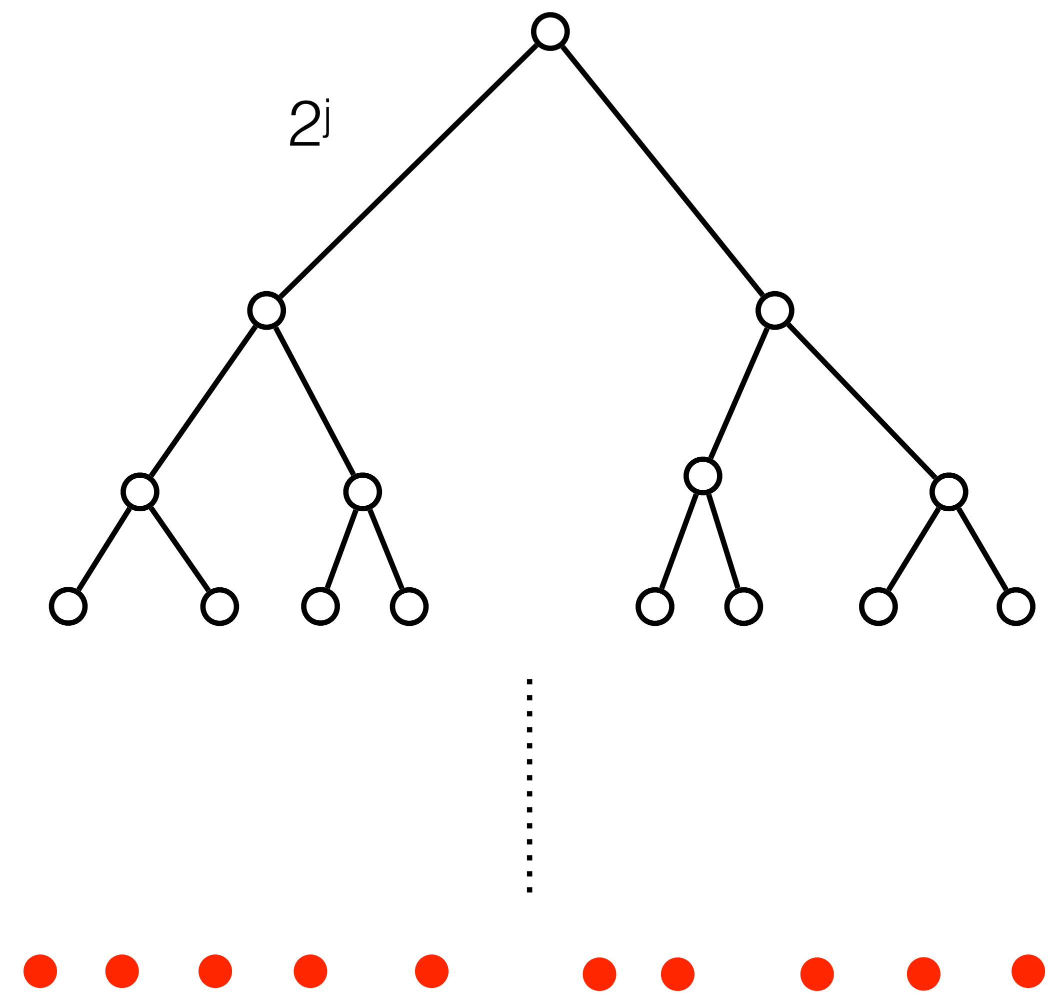
Proof Strategy:



1. Decompose OPT(T) into contributions from tree edges
2. Charge ALG to tree edges

Goal: ALG ≤ O(1) OPT(T) for any HST embedding T

Proof Strategy:

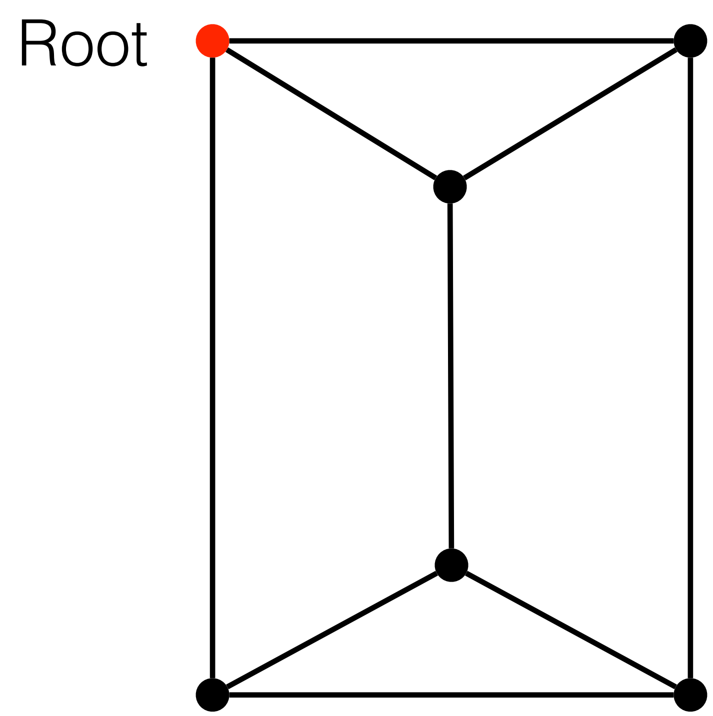


1. Decompose OPT(T) into contributions from tree edges
2. Charge ALG to tree edges
3. Use bounded-diameter property to argue no edge is overcharged

Outline

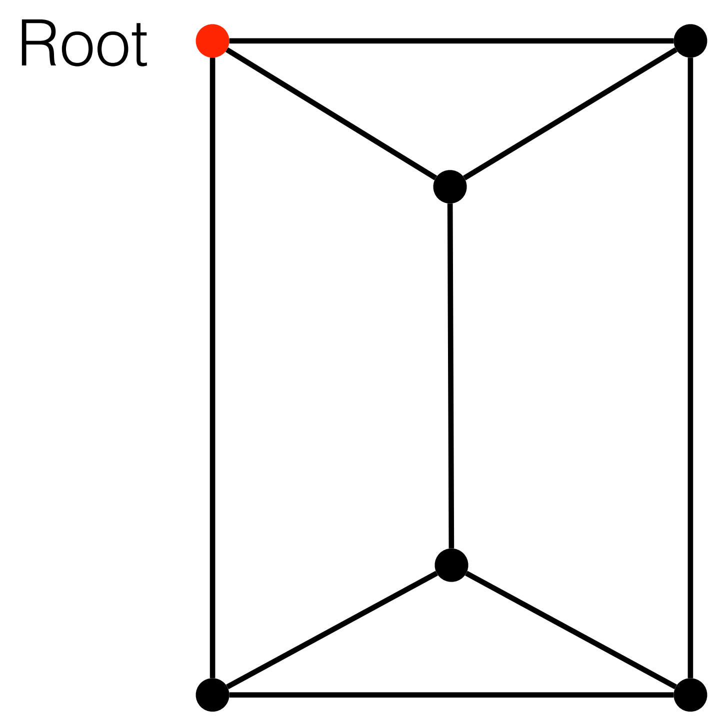
1. Overview of Analysis Framework ✔
2. Warm-Up: Steiner Tree
3. Rent-or-Buy
4. Steiner Network

Online Steiner Tree



- Terminals arrive over time
- Maintain min-cost subgraph connecting terminals to root

Online Steiner Tree

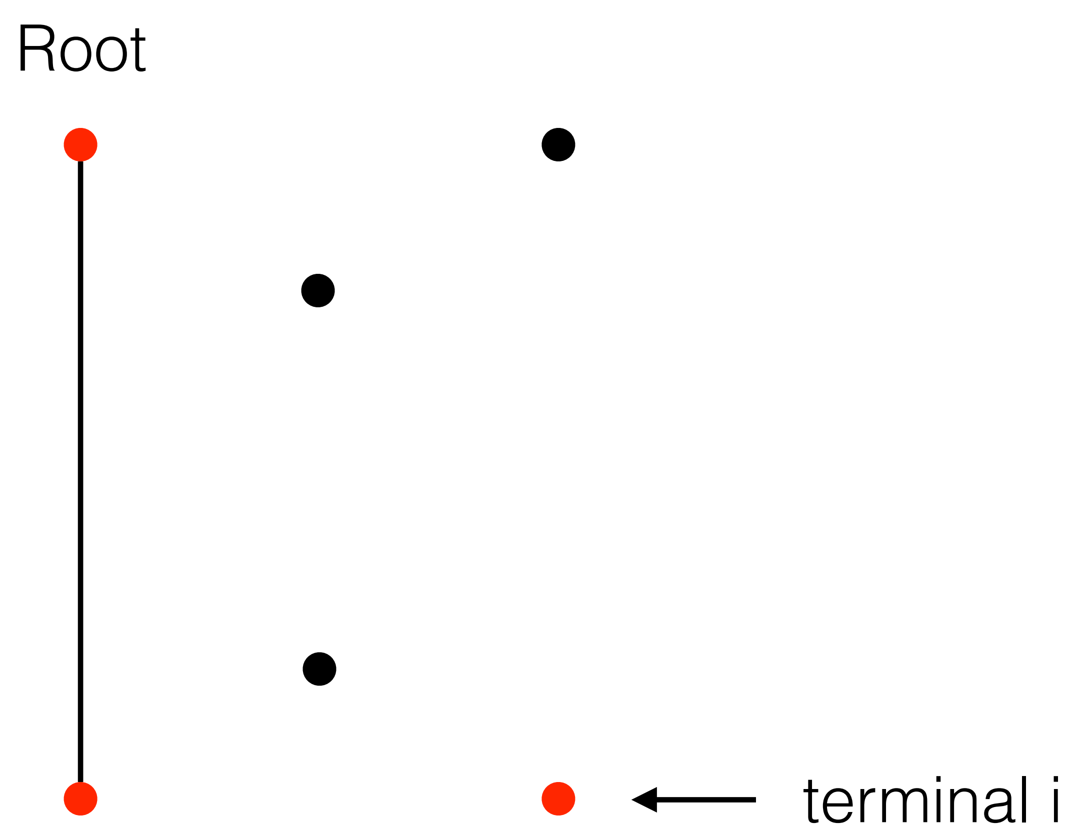


- Terminals arrive over time
- Maintain min-cost subgraph connecting terminals to root

We will show: greedy algorithm is O(log k)-competitive [Imase-Waxman 91]

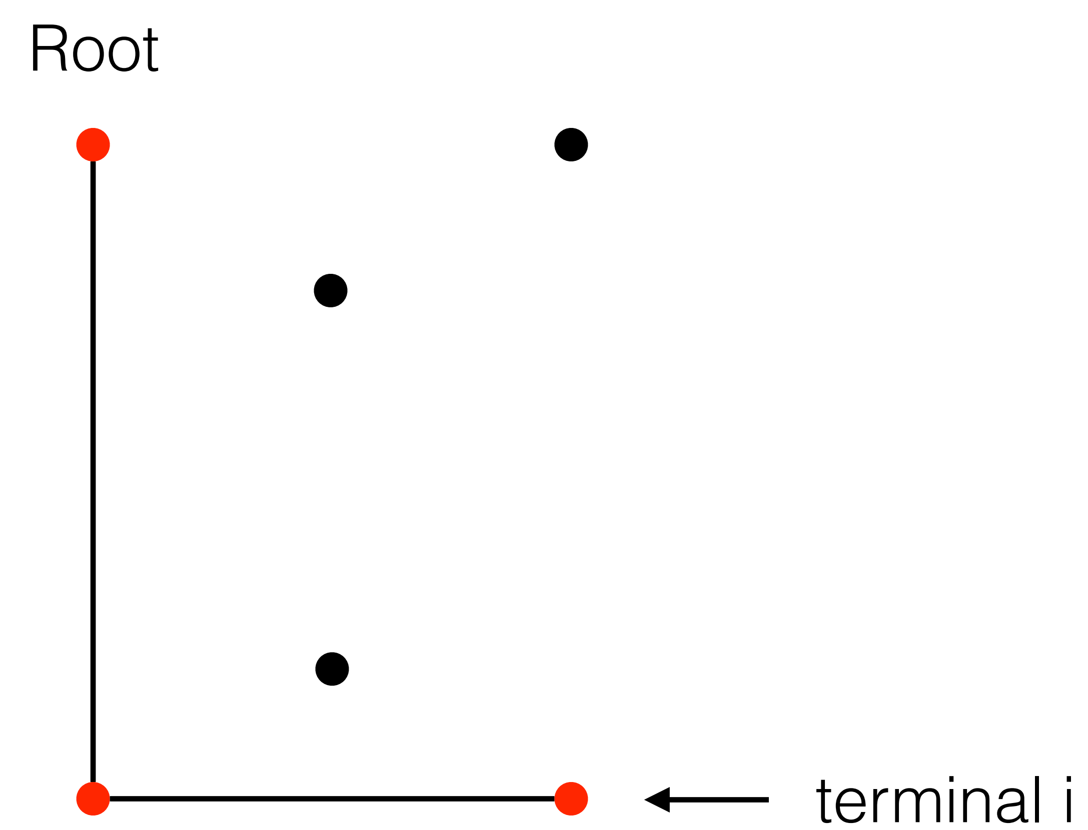
Greedy Algorithm

Connect current terminal to nearest previous terminal



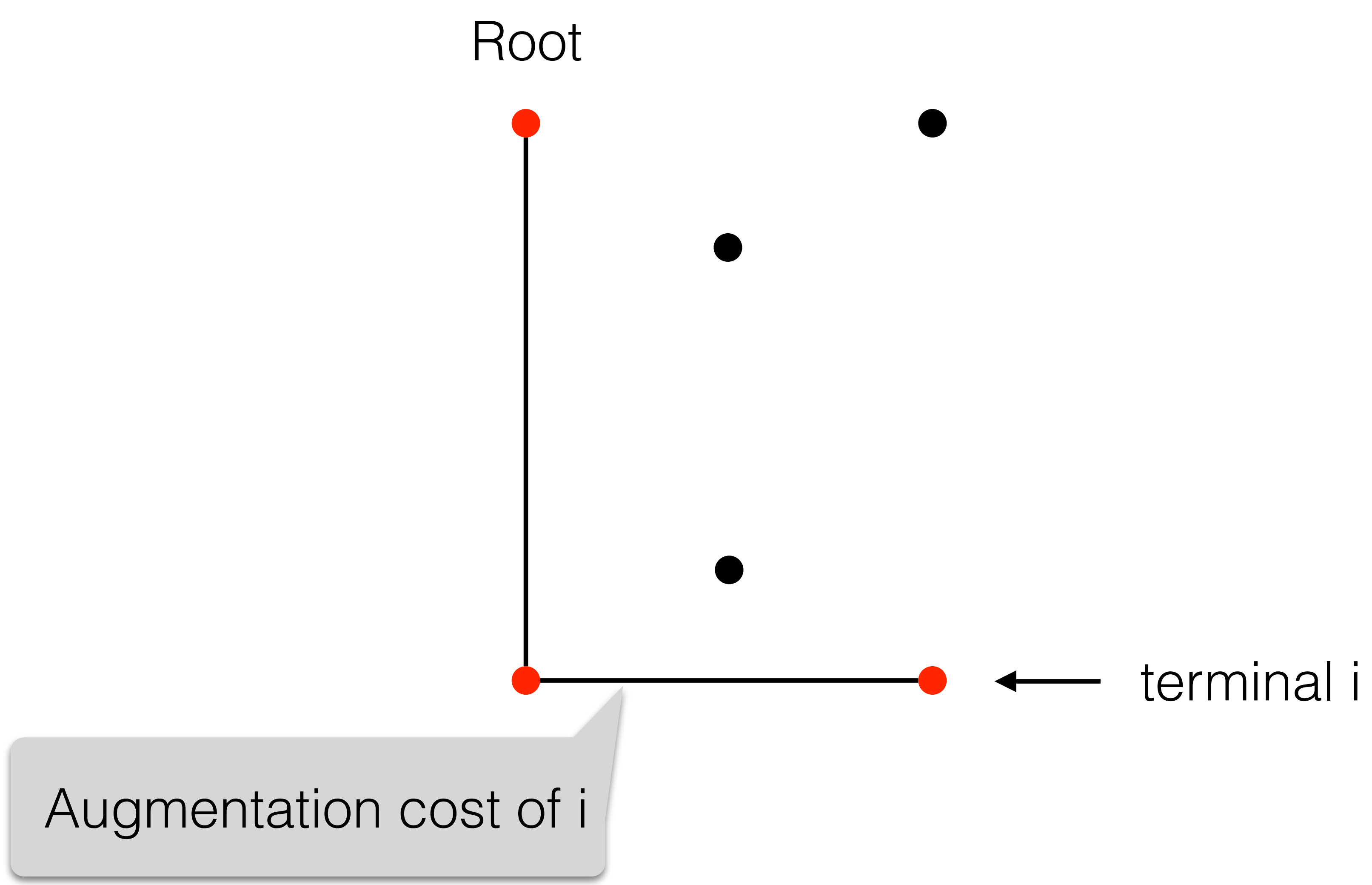
Greedy Algorithm

Connect current terminal to nearest previous terminal



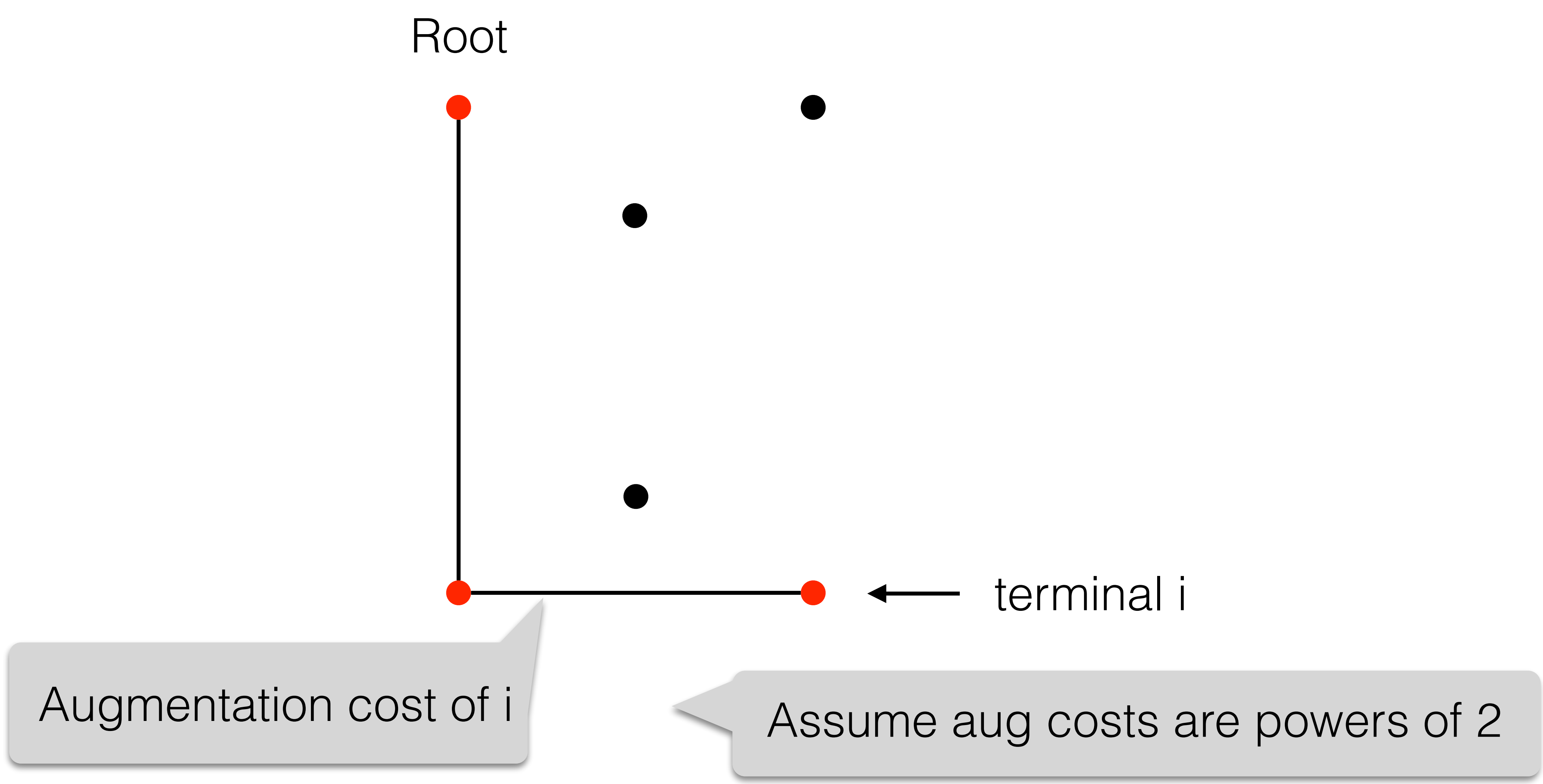
Greedy Algorithm

Connect current terminal to nearest previous terminal



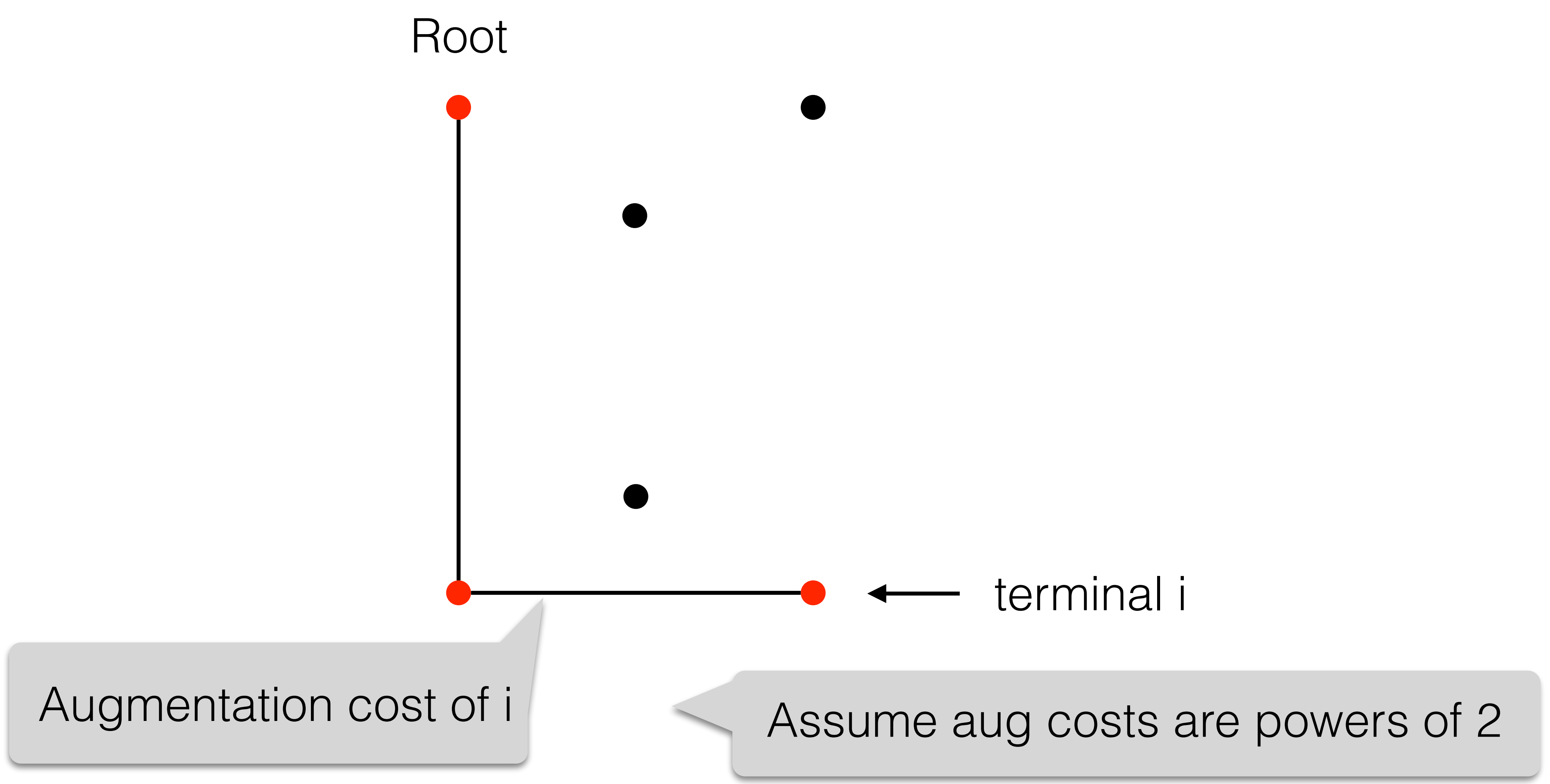
Greedy Algorithm

Connect current terminal to nearest previous terminal



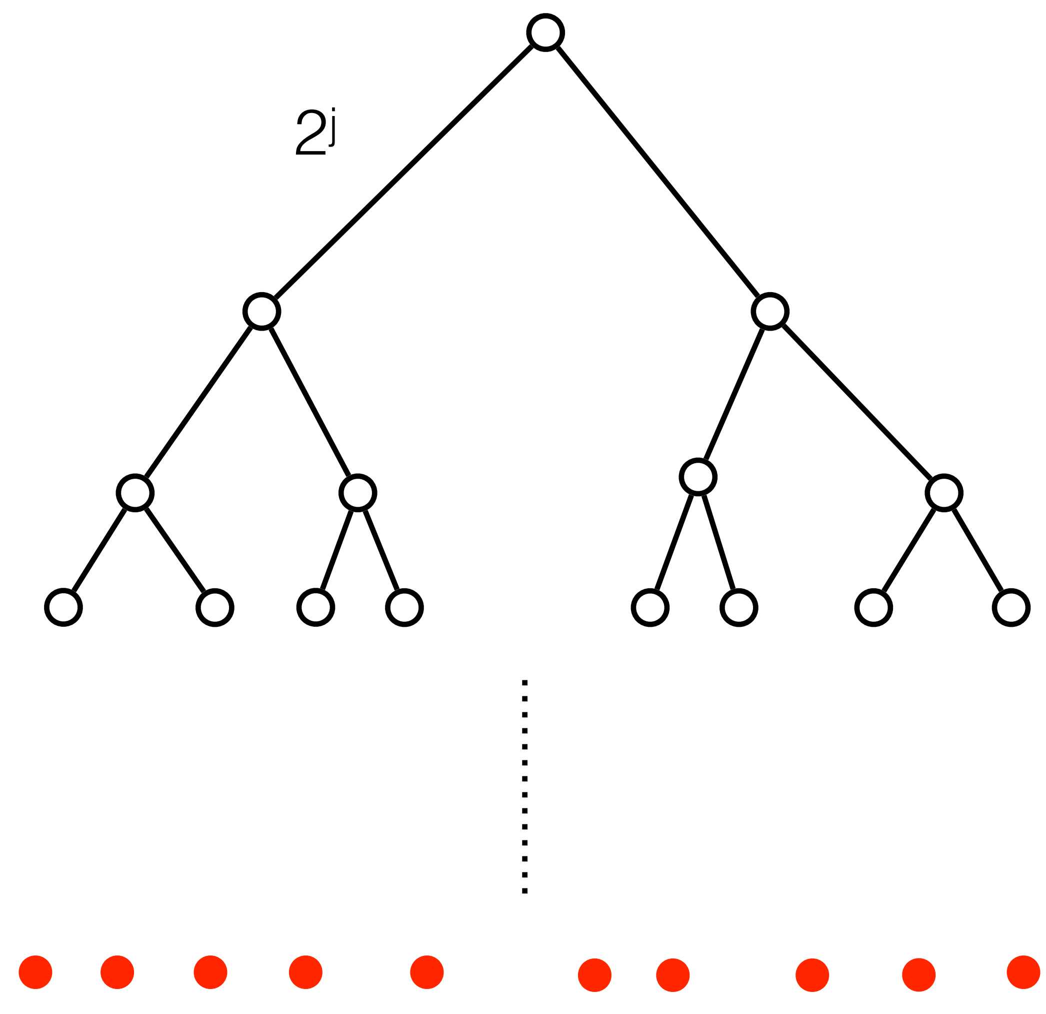
Greedy Algorithm

Connect current terminal to nearest previous terminal



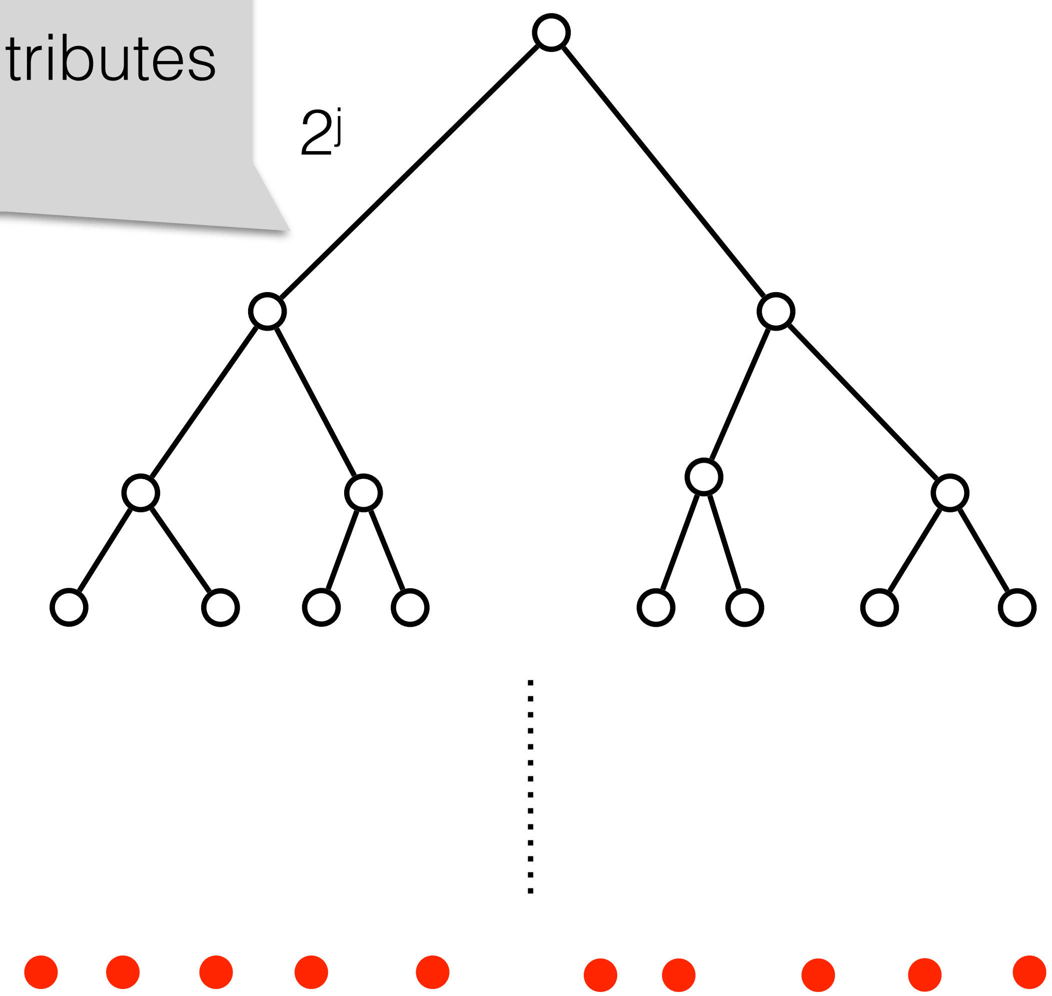
Claim: Total augmentation cost $\leq$ OPT(T)

Analysis

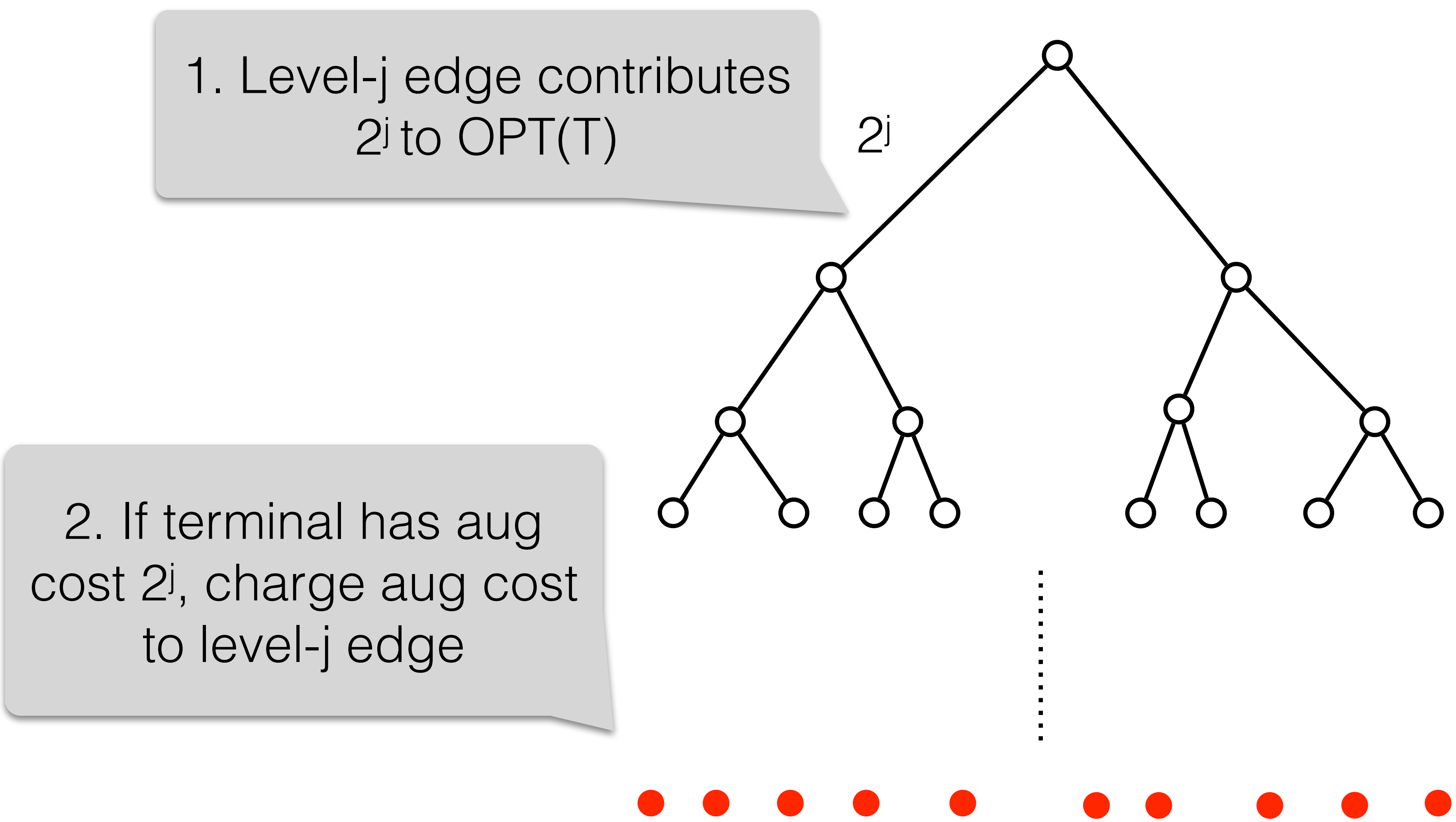


Analysis

1. Level-j edge contributes 2^j to OPT(T)



Analysis

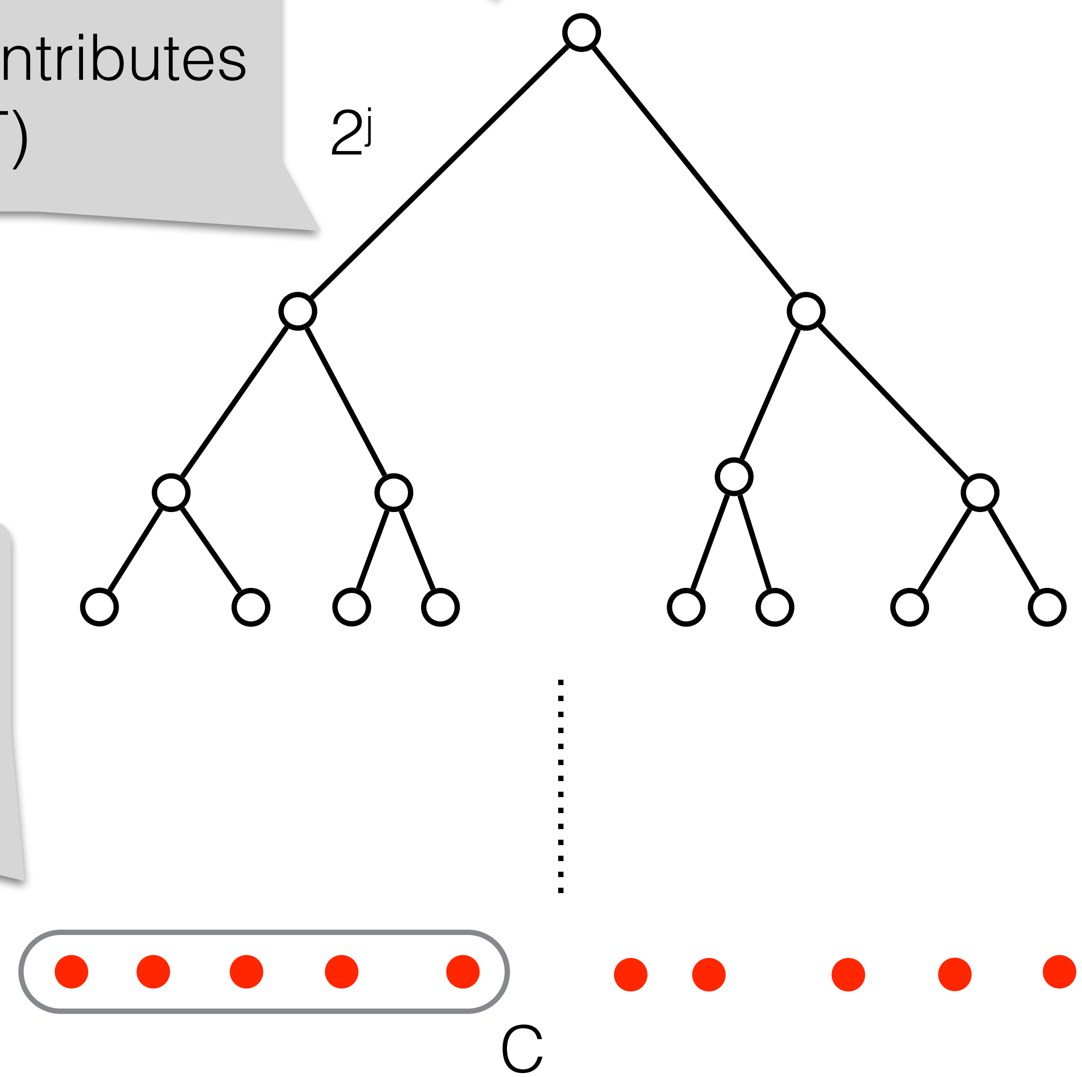

1. Level-j edge contributes 2^j to OPT(T)
2^j
2. If terminal has aug cost 2^j, charge aug cost to level-j edge

Analysis

3a. Charged 2^j for each terminal in C with aug cost 2^j

1. Level-j edge contributes 2^j to OPT(T)

2. If terminal has aug cost 2^j, charge aug cost to level-j edge

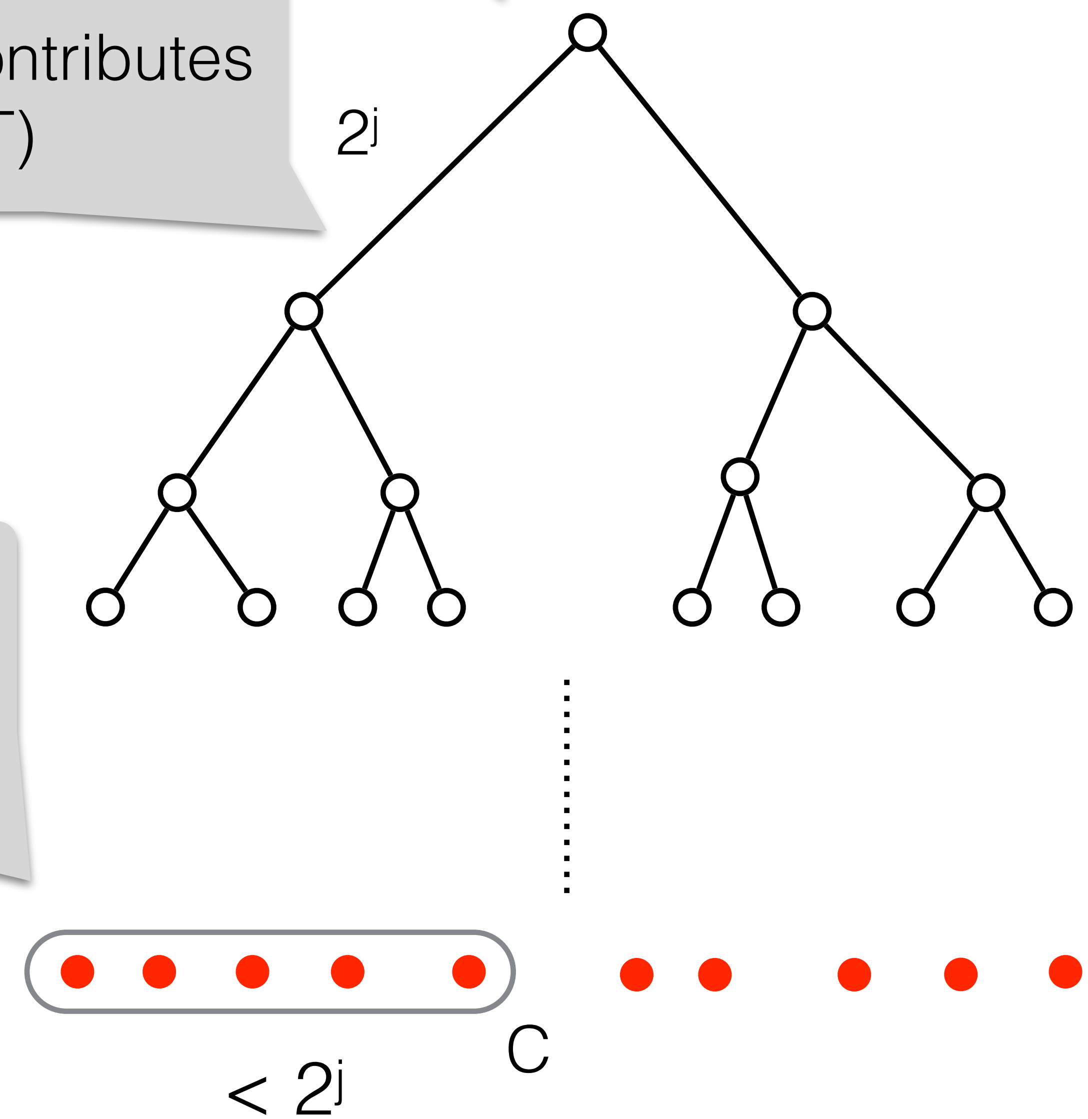


Analysis

3a. Charged 2^j for each terminal in C with aug cost 2^j

1. Level-j edge contributes 2^j to OPT(T)

2. If terminal has aug cost 2^j, charge aug cost to level-j edge



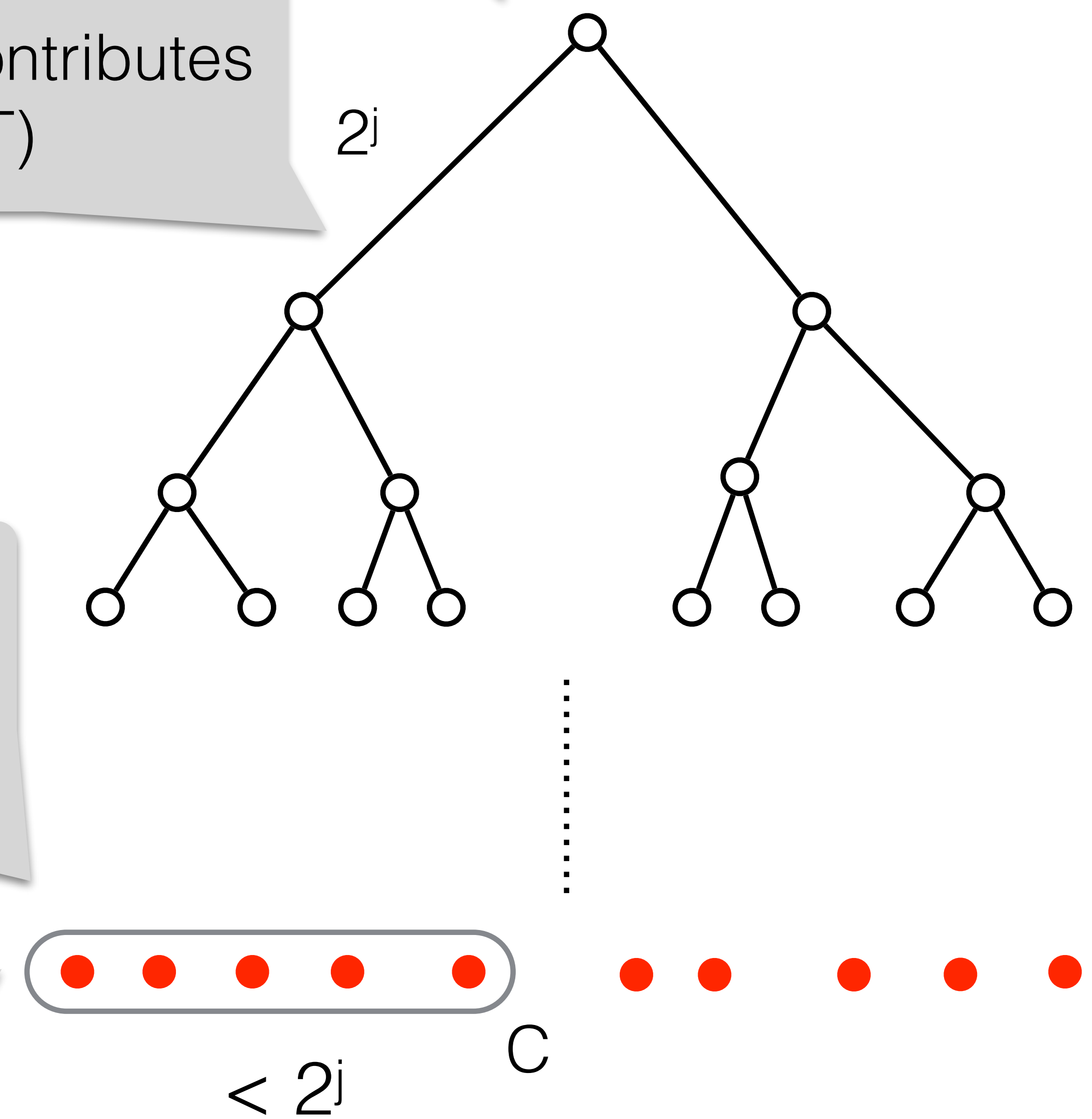
Analysis

3a. Charged 2^j for each terminal in C with aug cost 2^j

1. Level-j edge contributes 2^j to OPT(T)

2. If terminal has aug cost 2^j, charge aug cost to level-j edge

3b. At most one terminal in C with aug cost 2^j



Lemma: Greedy $\leq$ OPT(T) for any HST embedding T

Lemma: Greedy $\leq$ OPT(T) for any HST embedding T

Theorem [Imase-Waxman 91]:
Greedy is O(log k)-competitive for online Steiner tree

Lemma: Greedy ≤ OPT(T) for any HST embedding T

Theorem [Imase-Waxman 91]:
Greedy is O(log k)-competitive for online Steiner tree

We also get a simpler analysis for the online Steiner forest algorithm of Berman-Coulston.

Outline

1. Overview of Analysis Framework ✔
2. Warm-Up: Steiner Tree ✔
3. Rent-or-Buy
4. Steiner Network

Rent-or-Buy

Rent-or-Buy

- Initially, given root and parameter $M \geq 1$

Rent-or-Buy

- Initially, given root and parameter $M \geq 1$
- Maintain a connected subgraph H containing root

Rent-or-Buy

- Initially, given root and parameter $M \geq 1$
- Maintain a connected subgraph H containing root
- When given terminal i:
  - "buy" edges (add them to H), and
  - "rent" edges Q_i so that $H \cup Q_i$ connects i to r

Rent-or-Buy

- Initially, given root and parameter $M \geq 1$
- Maintain a connected subgraph H containing root
- When given terminal i:
  - “buy” edges (add them to H), and
  - “rent” edges Q_i so that $H \cup Q_i$ connects i to r

Cost:

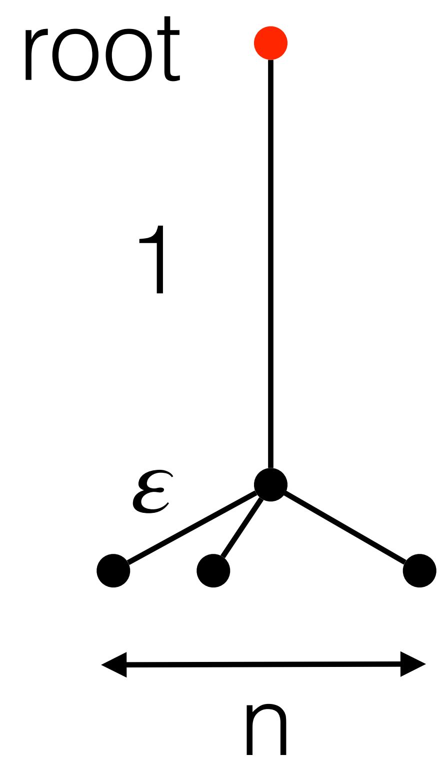
Rent-or-Buy

- Initially, given root and parameter $M \geq 1$
- Maintain a connected subgraph H containing root
- When given terminal i:
  - "buy" edges (add them to H), and
  - "rent" edges Q_i so that H U Q_i connects i to r

Cost: $\underbrace{Mc(H)}_{\text{Buy cost}} + \sum_i \underbrace{c(Q_i)}_{\text{Rent cost of i}}$

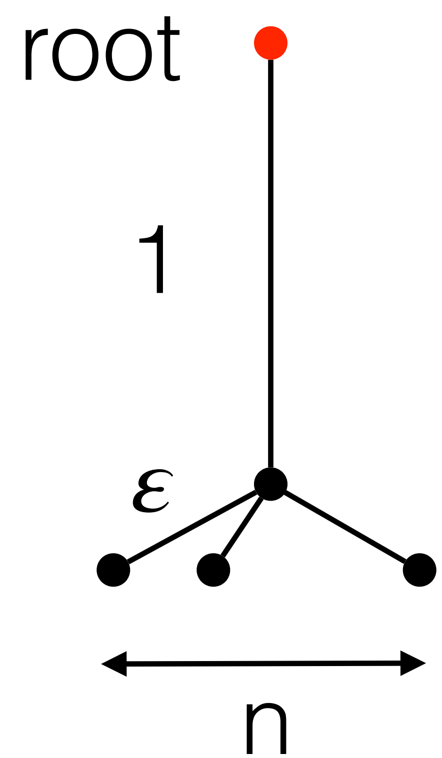
Example

Tree instance



Example

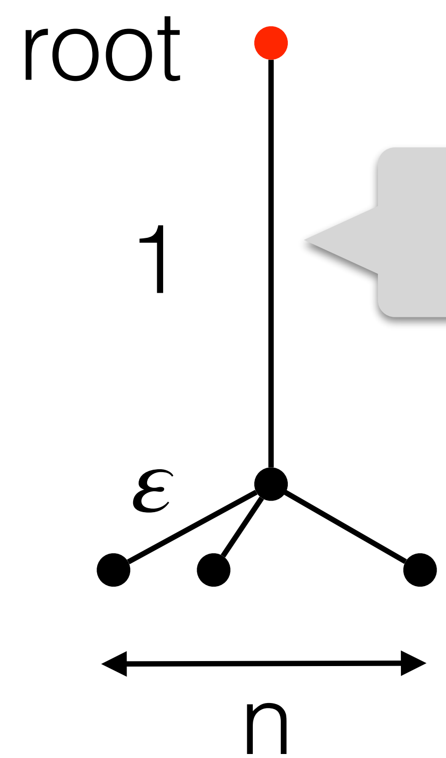
Tree instance



Terminals are leaves

Example

Tree instance

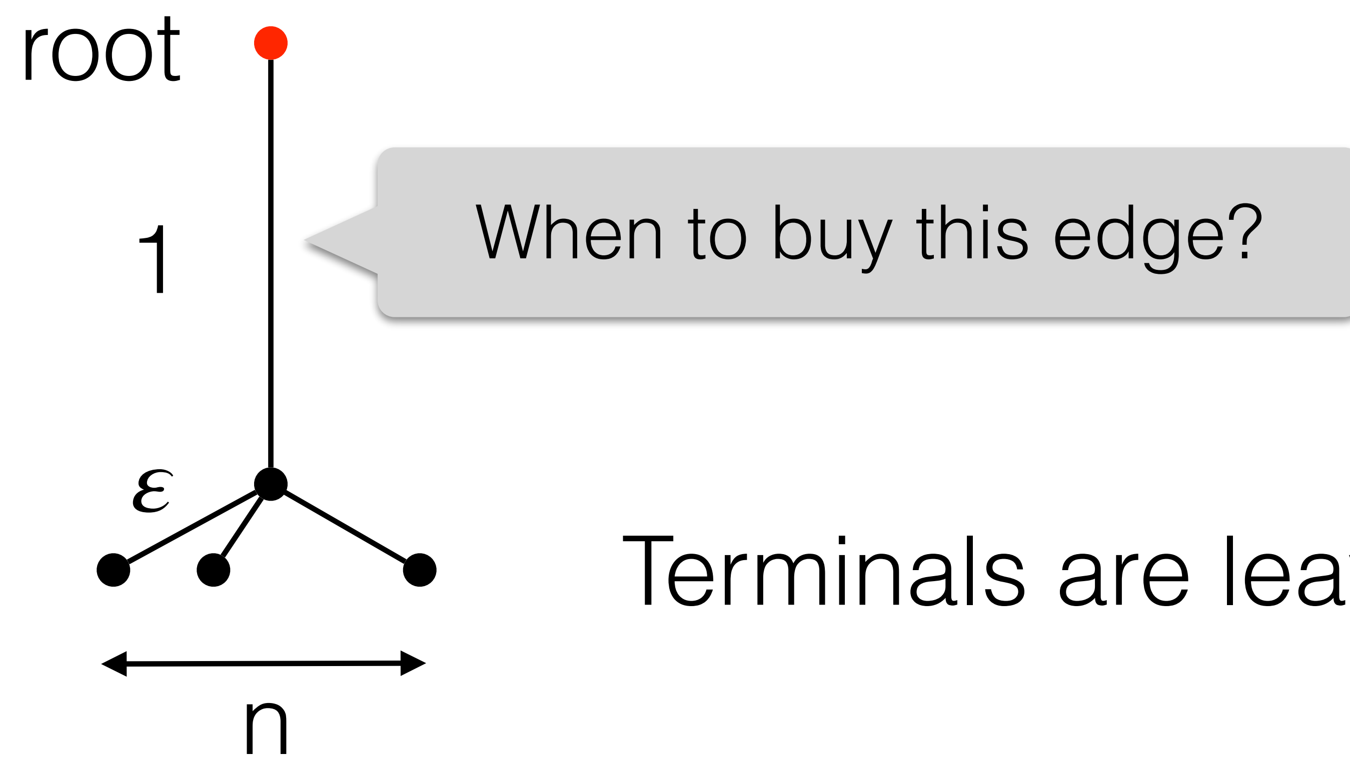


When to buy this edge?

Terminals are leaves

Example

Tree instance

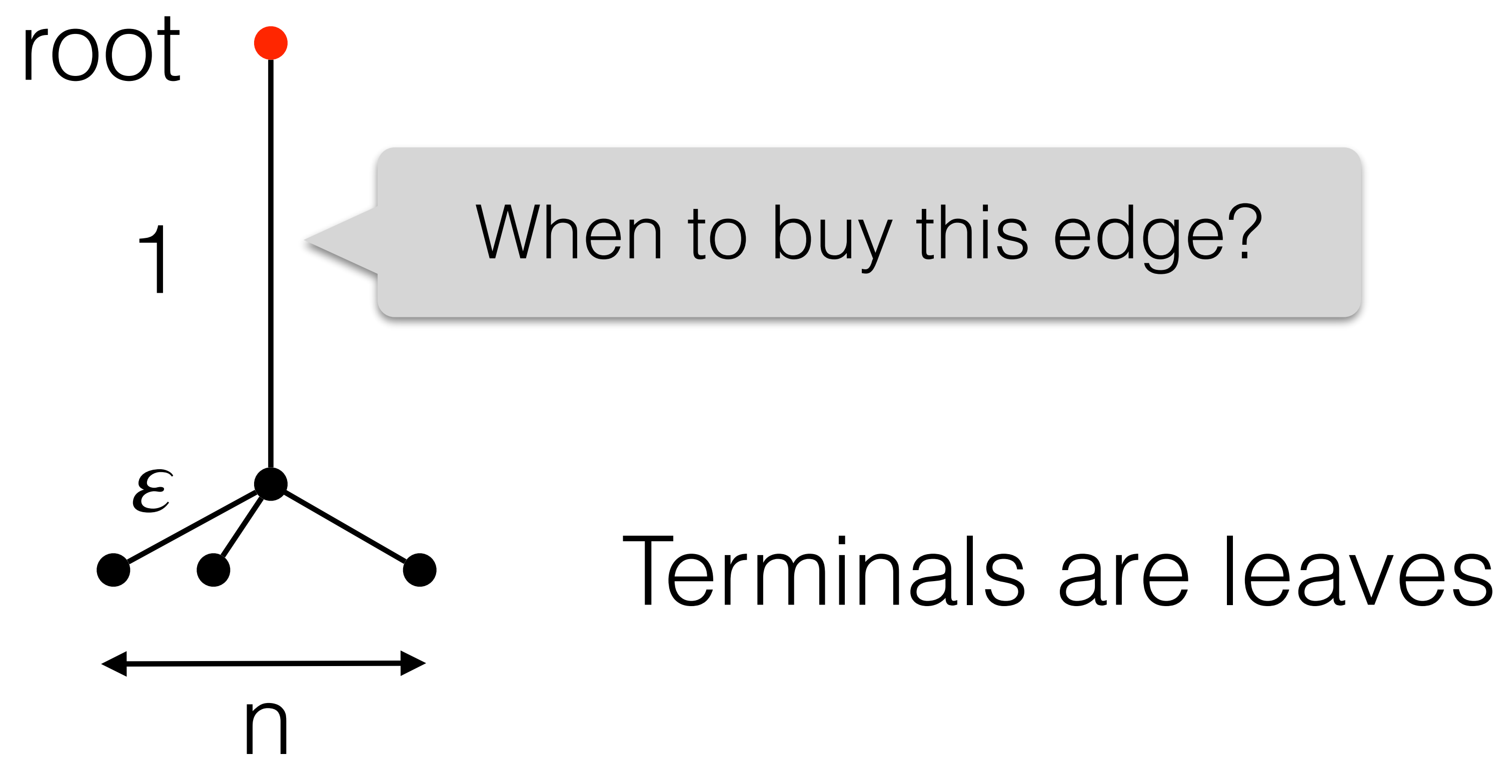


Terminals are leaves

OPT = min {#terminals, M}

Example

Tree instance



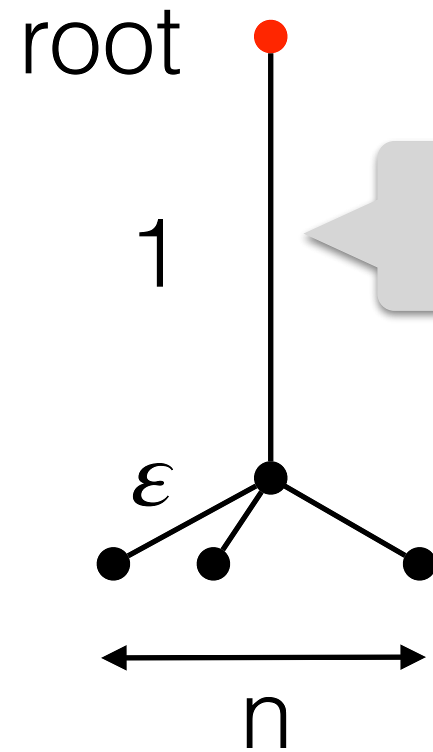
OPT = min {#terminals, M}

<u>Online Algorithm</u>

First M terminals: Rent
(M+1)th terminal: Buy

Example

Tree instance



When to buy this edge?

Terminals are leaves

OPT = min {#terminals, M}

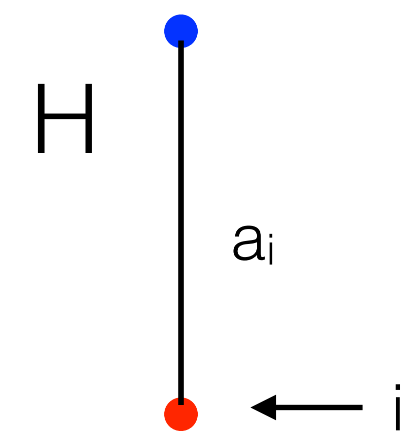
<u>Online Algorithm</u>

First M terminals: Rent
(M+1)th terminal: Buy

Buy when sufficient rent cost to pay for buy cost (break-even rule)

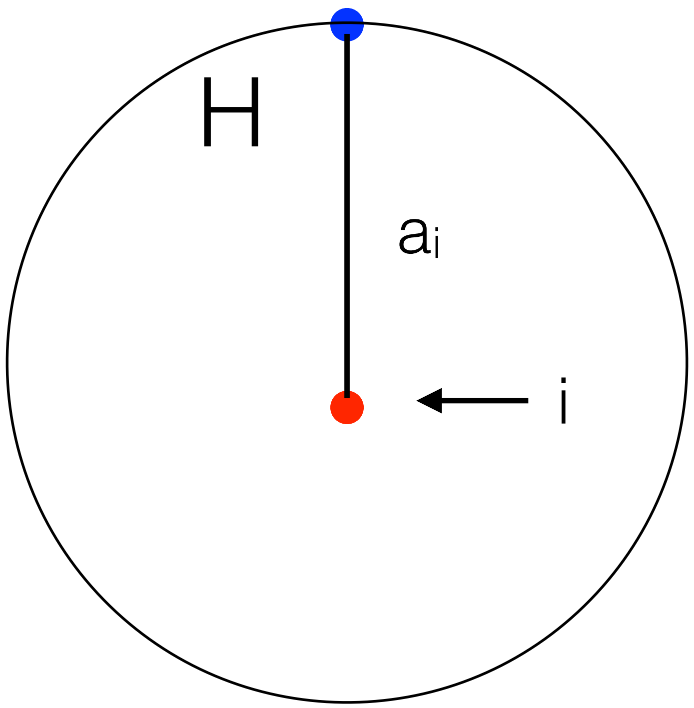
<u>Algorithm:</u>

- When a terminal i arrives:
    - Let e be shortest edge to H and a_i its length
    - If $\geq$ M terminals with rent cost $\geq a_i$ within distance a_i, buy e at cost Ma_i
    - Otherwise, rent e at cost a_i



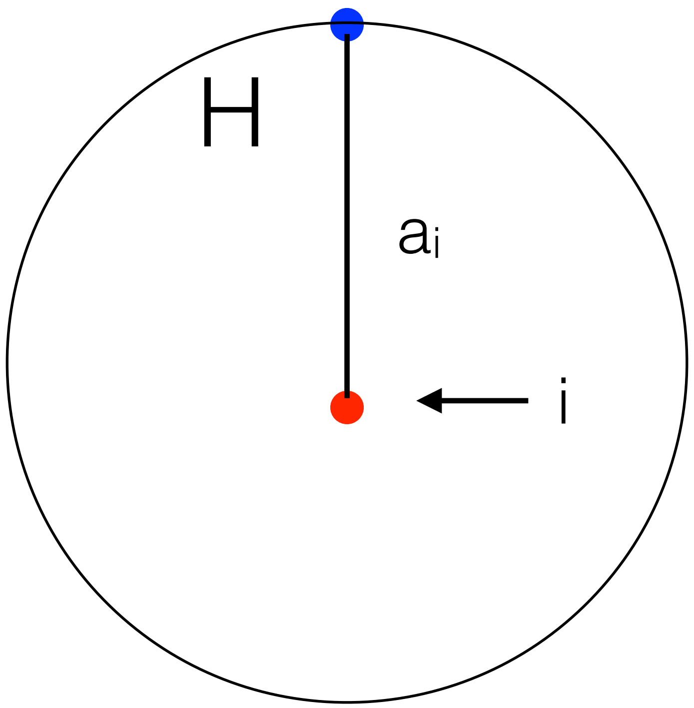
Algorithm:

- When a terminal i arrives:
  - Let e be shortest edge to H and a_i its length
  - If ≥ M terminals with rent cost ≥ a_i within distance a_i, buy e at cost Ma_i
  - Otherwise, rent e at cost a_i



Algorithm:

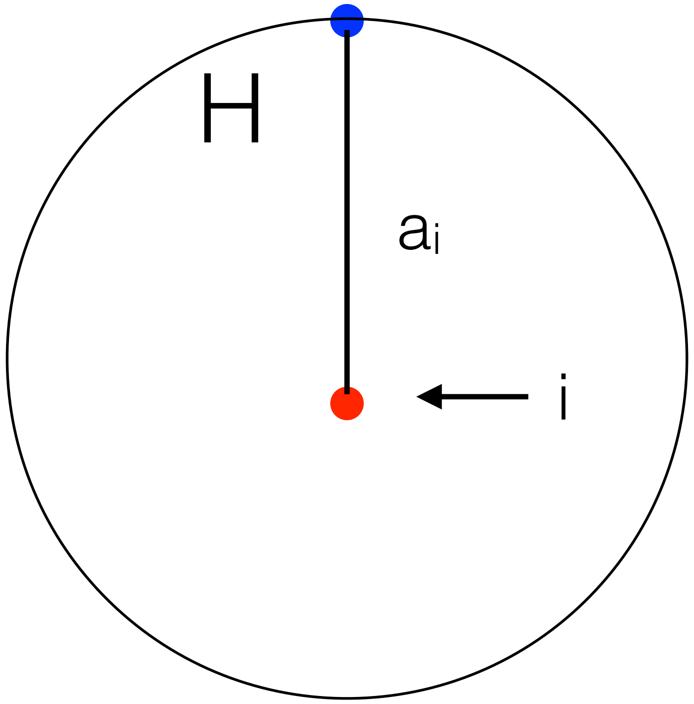
- When a terminal i arrives:
    - Let e be shortest edge to H and a_i its length
    - If $\geq M$ terminals with rent cost $\geq a_i$ within distance a_i, buy e at cost Ma_i
    - Otherwise, rent e at cost a_i



Analysis:

Algorithm:

- When a terminal i arrives:
  - Let e be shortest edge to H and a_i its length
  - If $\geq$ M terminals with rent cost $\geq a_i$ within distance a_i, buy e at cost Ma_i
  - Otherwise, rent e at cost a_i

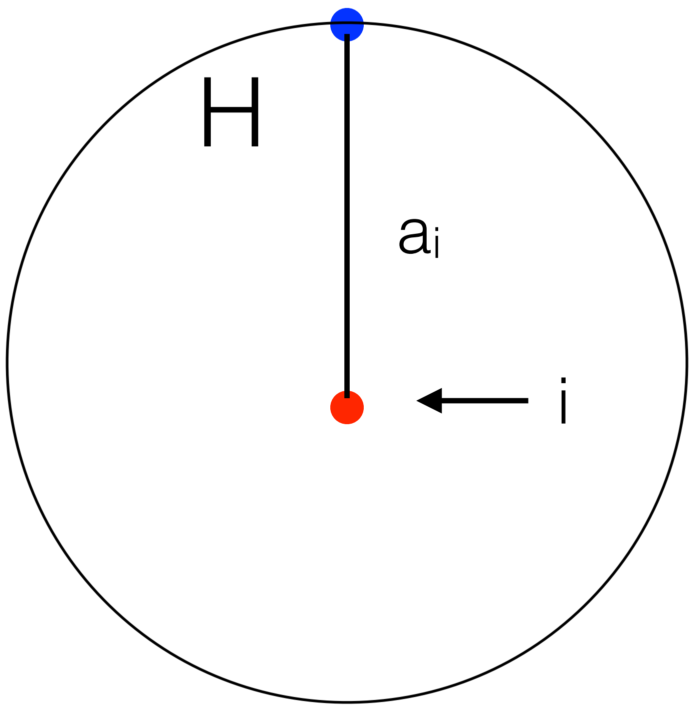


Analysis:

Lemma: Buy cost $\leq$ O(1) Rent cost

Algorithm:

- When a terminal i arrives:
  - Let e be shortest edge to H and a_i its length
  - If $\geq$ M terminals with rent cost $\geq a_i$ within distance a_i, buy e at cost Ma_i
  - Otherwise, rent e at cost a_i



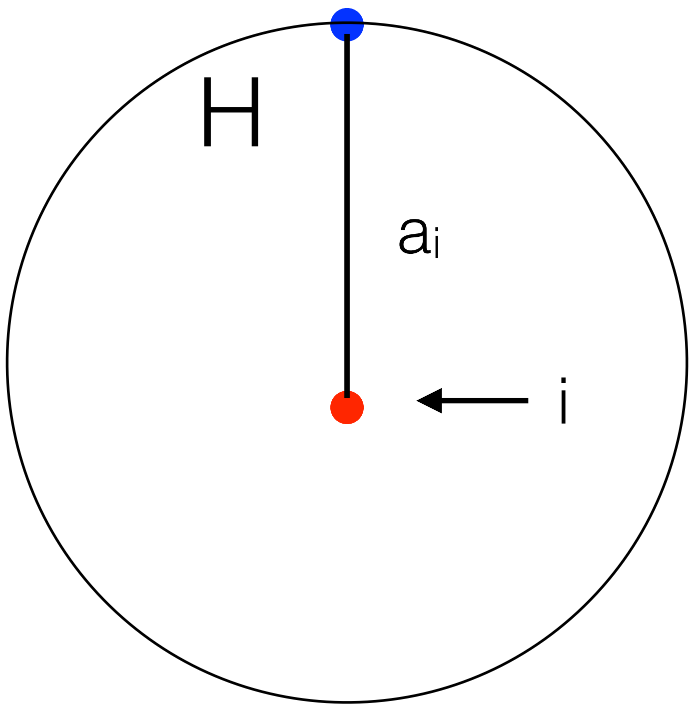
Analysis:

Lemma: Buy cost $\leq$ O(1) Rent cost

Lemma: Rent cost $\leq$ O(1)OPT(T) for any HST embedding T

Algorithm:

- When a terminal i arrives:
  - Let e be shortest edge to H and a_i its length
  - If $\geq$ M terminals with rent cost $\geq a_i$ within distance a_i, buy e at cost Ma_i
  - Otherwise, rent e at cost a_i



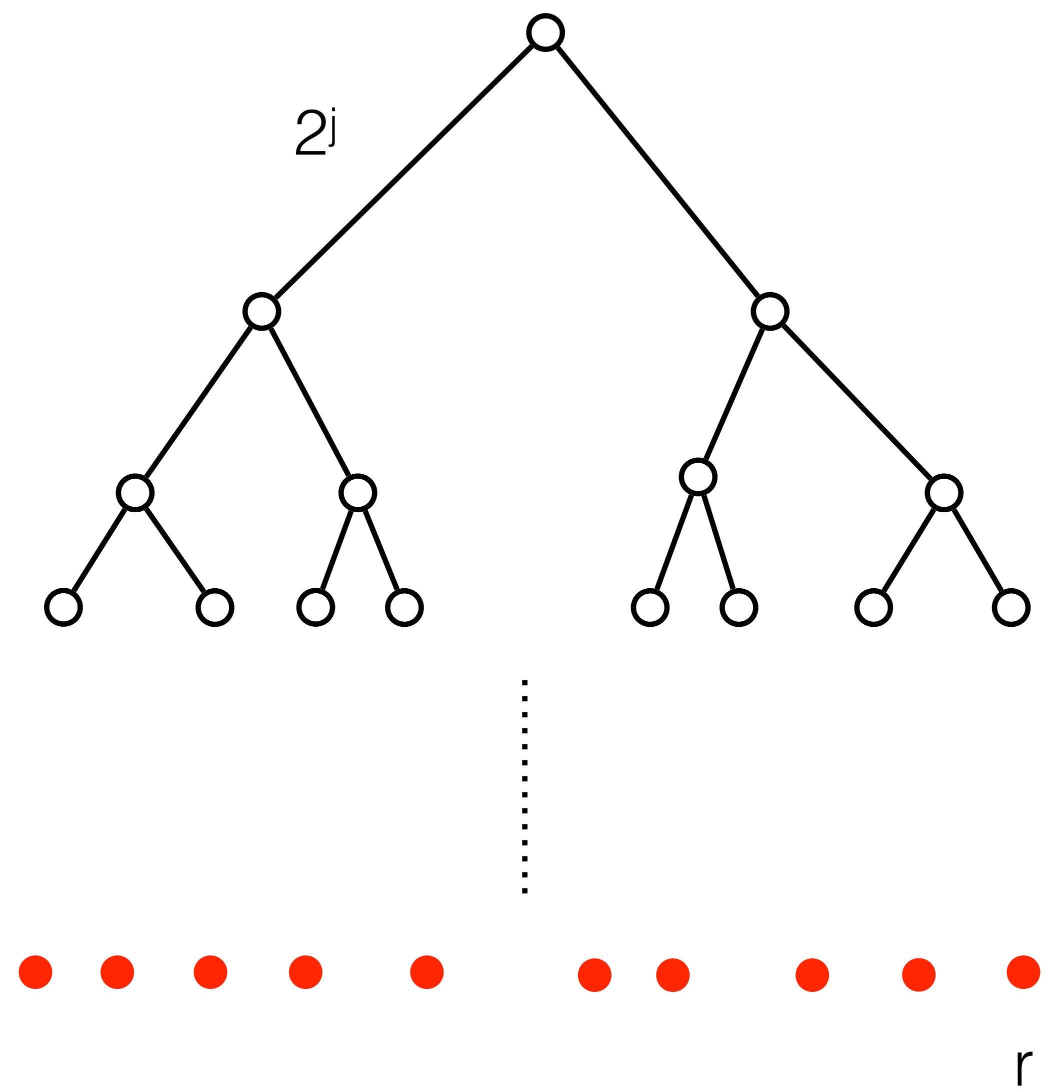
Analysis:

Lemma: Buy cost $\leq$ O(1) Rent cost

Lemma: Rent cost $\leq$ O(1)OPT(T) for any HST embedding T

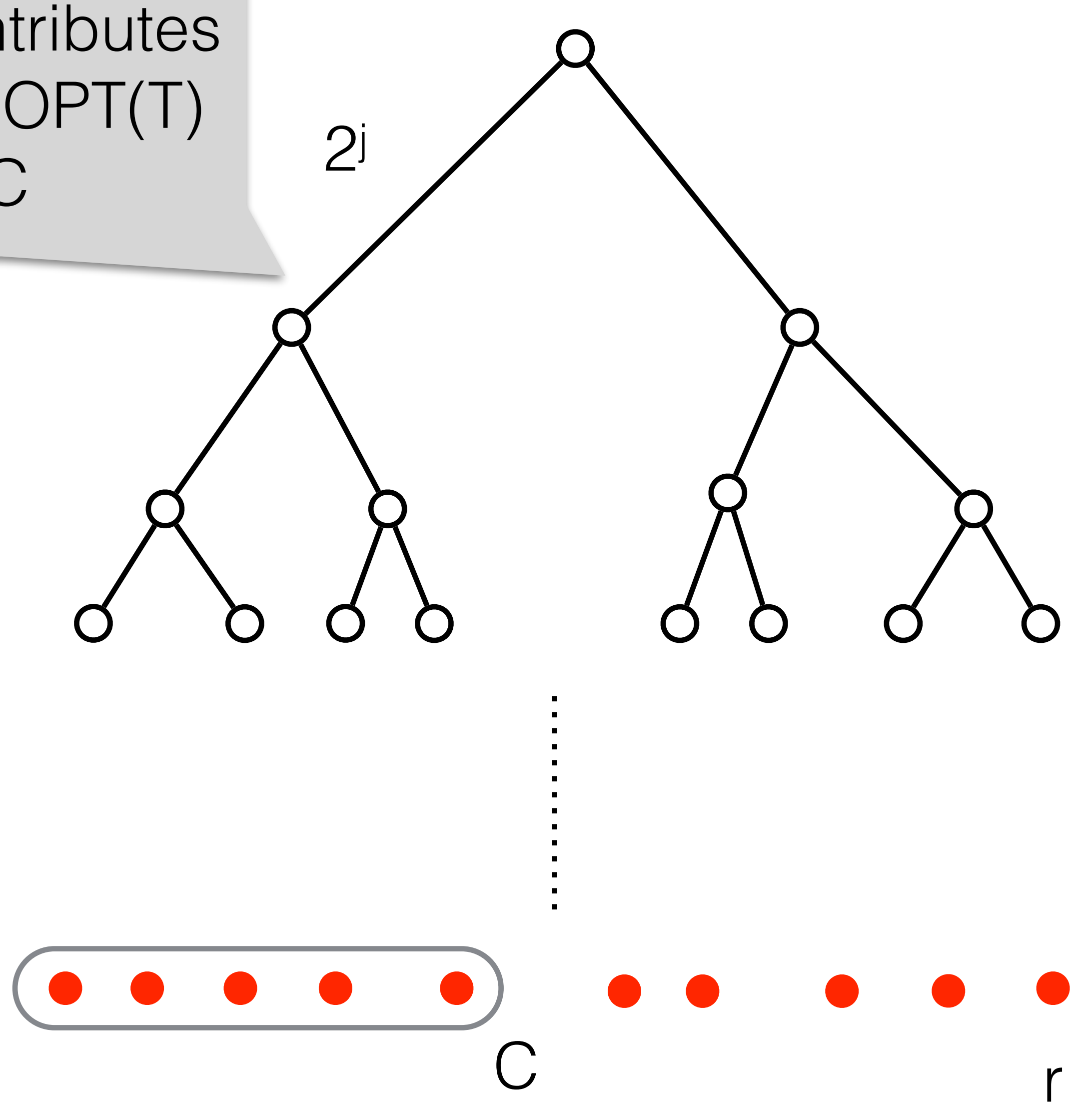
Theorem: ALG is O(log k)-competitive for ROB

Analysis



Analysis

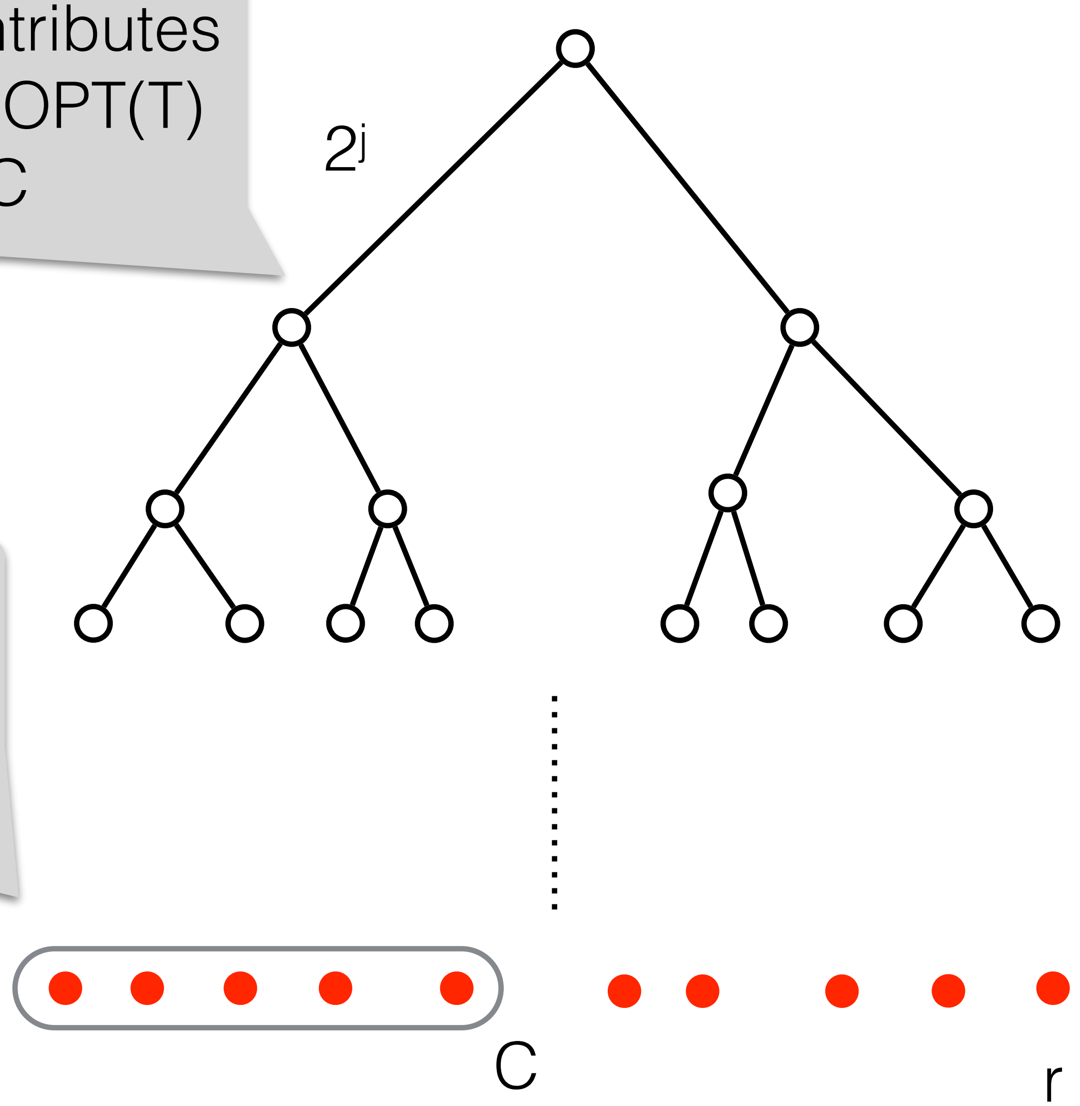
1. Level-j edge contributes $2^j \min\{M, |C|\}$ to OPT(T) if r not in C



Analysis

1. Level-j edge contributes $2^j \min\{M, |C|\}$ to OPT(T) if r not in C

2. If terminal has rent cost 2^j, charge rent cost to level-j edge

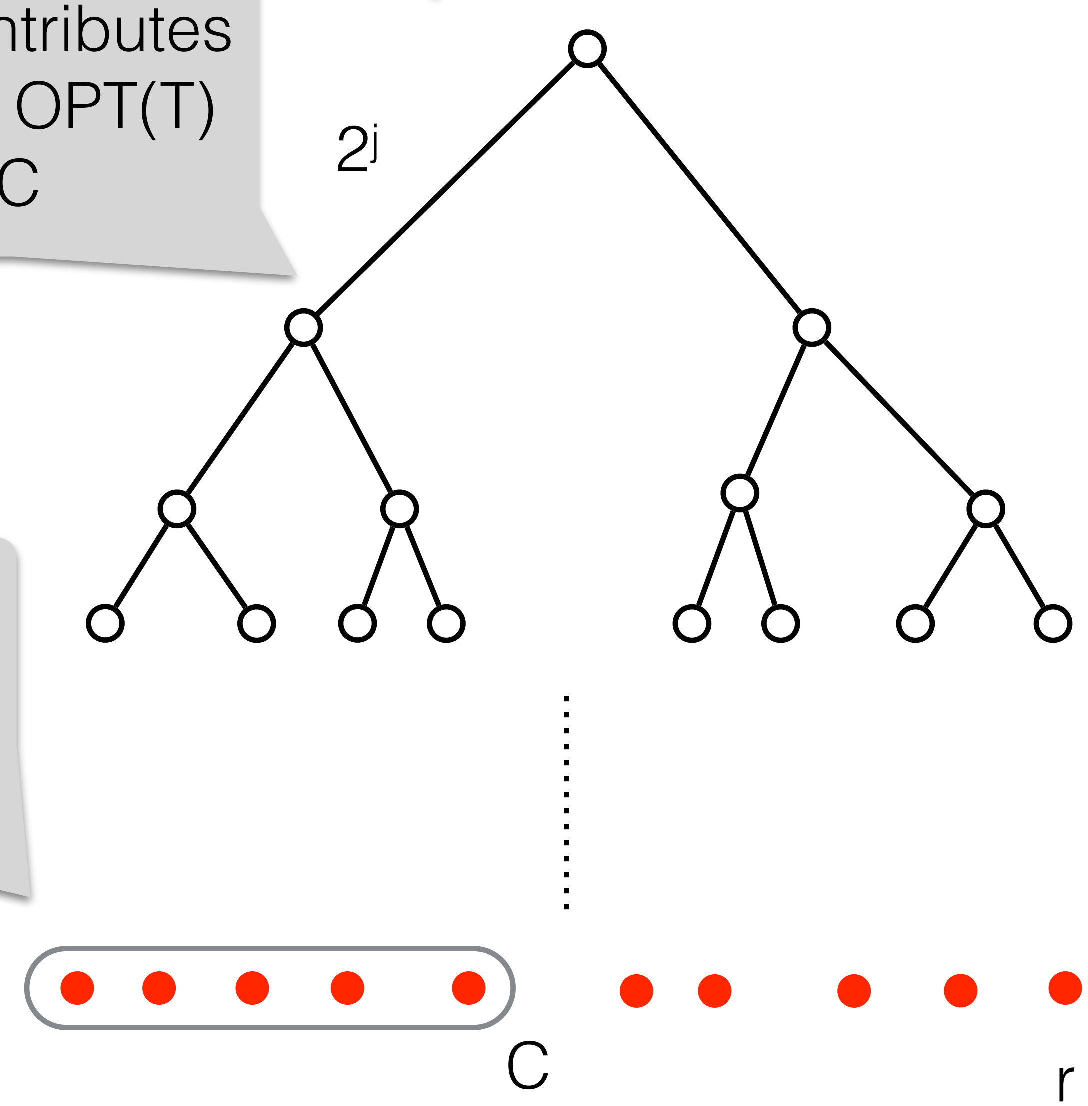


Analysis

3a. Charged 2^j for each terminal in C with rent cost 2^j

1. Level-j edge contributes $2^j \min\{M, |C|\}$ to OPT(T) if r not in C

2. If terminal has rent cost 2^j, charge rent cost to level-j edge

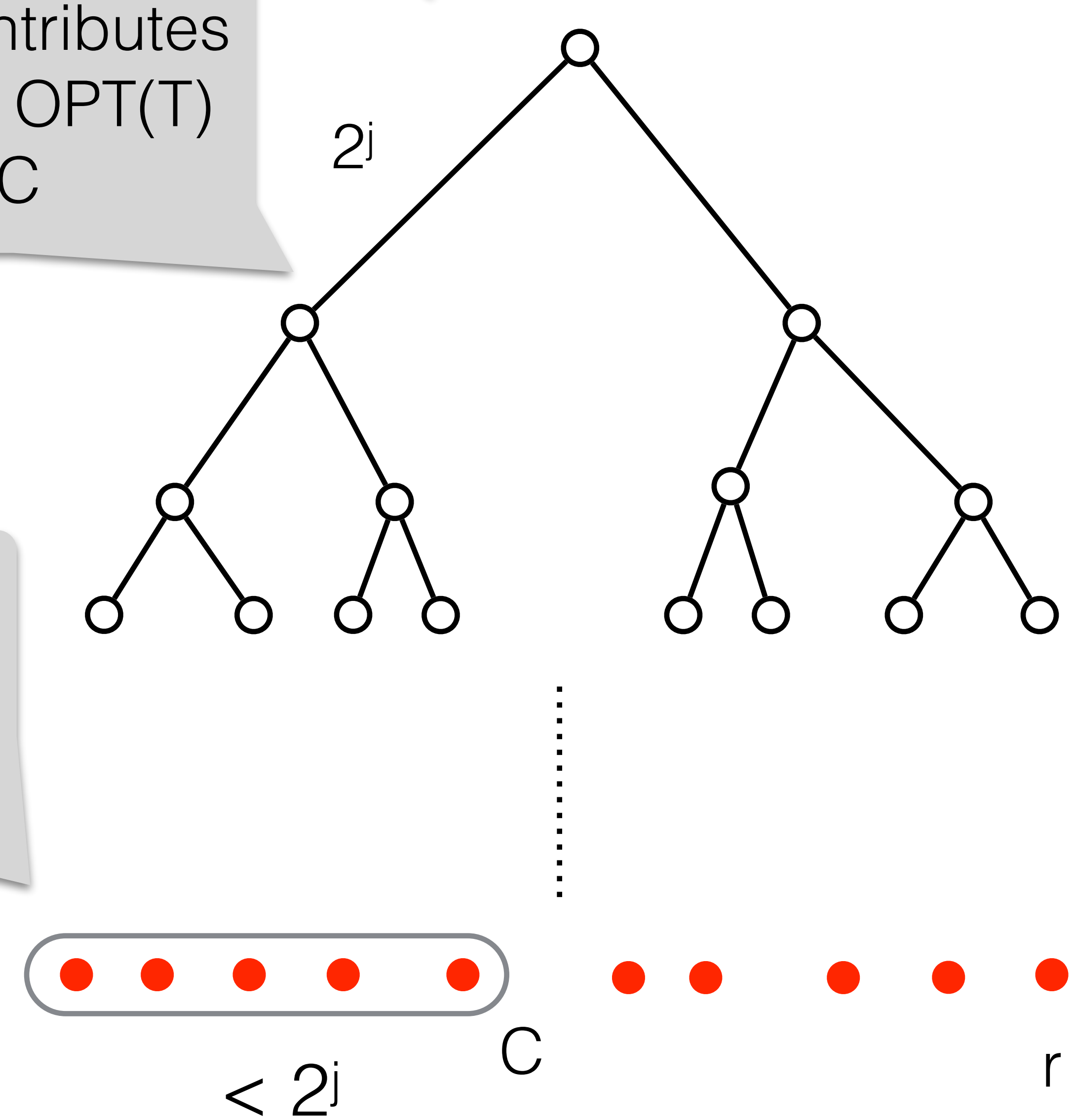


Analysis

3a. Charged 2^j for each terminal in C with rent cost 2^j

1. Level-j edge contributes $2^j \min\{M, |C|\}$ to OPT(T) if r not in C

2. If terminal has rent cost 2^j, charge rent cost to level-j edge



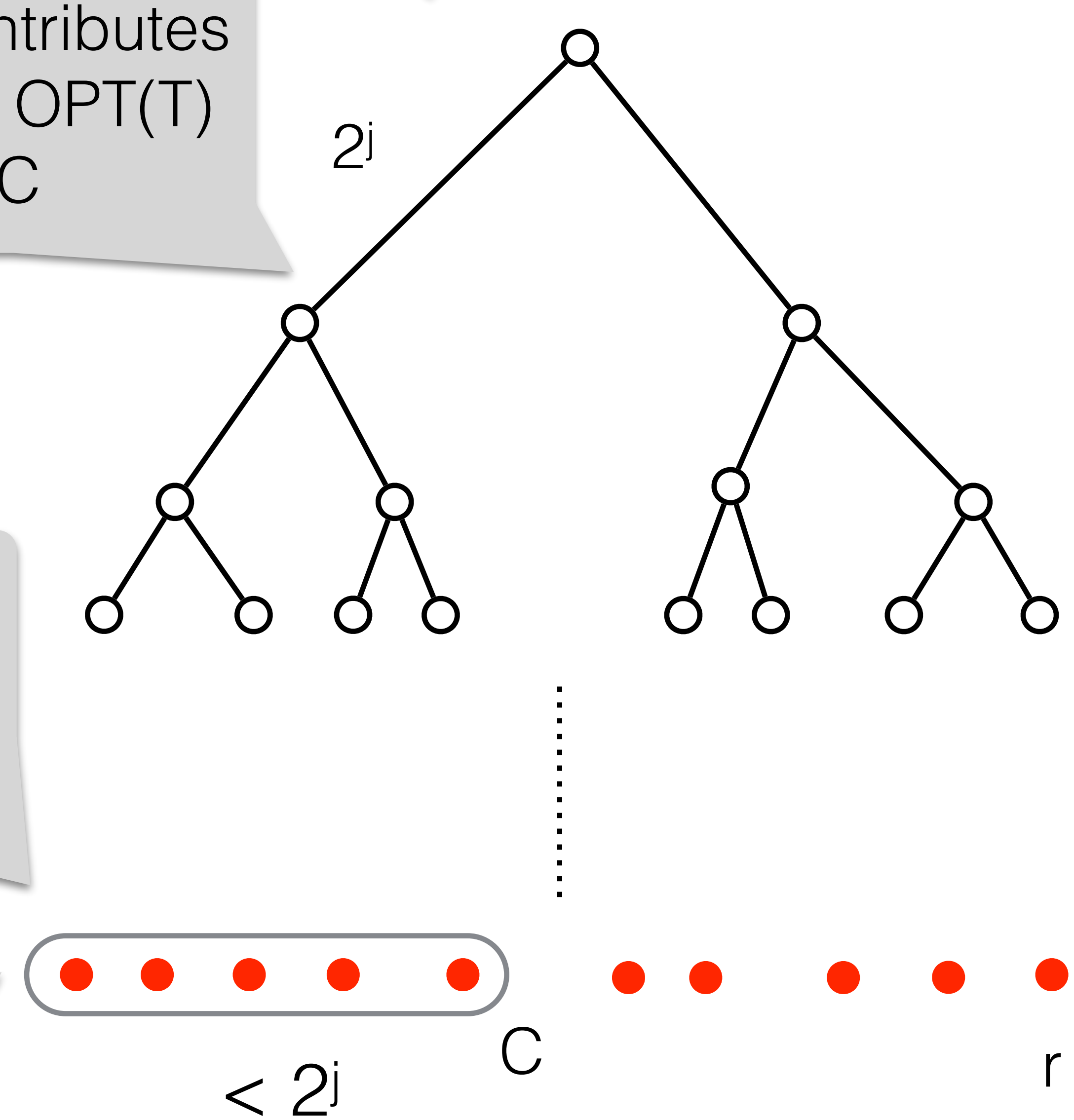
Analysis

3a. Charged 2^j for each terminal in C with rent cost 2^j

1. Level-j edge contributes $2^j \min\{M, |C|\}$ to OPT(T) if r not in C

2. If terminal has rent cost 2^j, charge rent cost to level-j edge

3b. At most M terminals in C with rent cost 2^j



Analysis

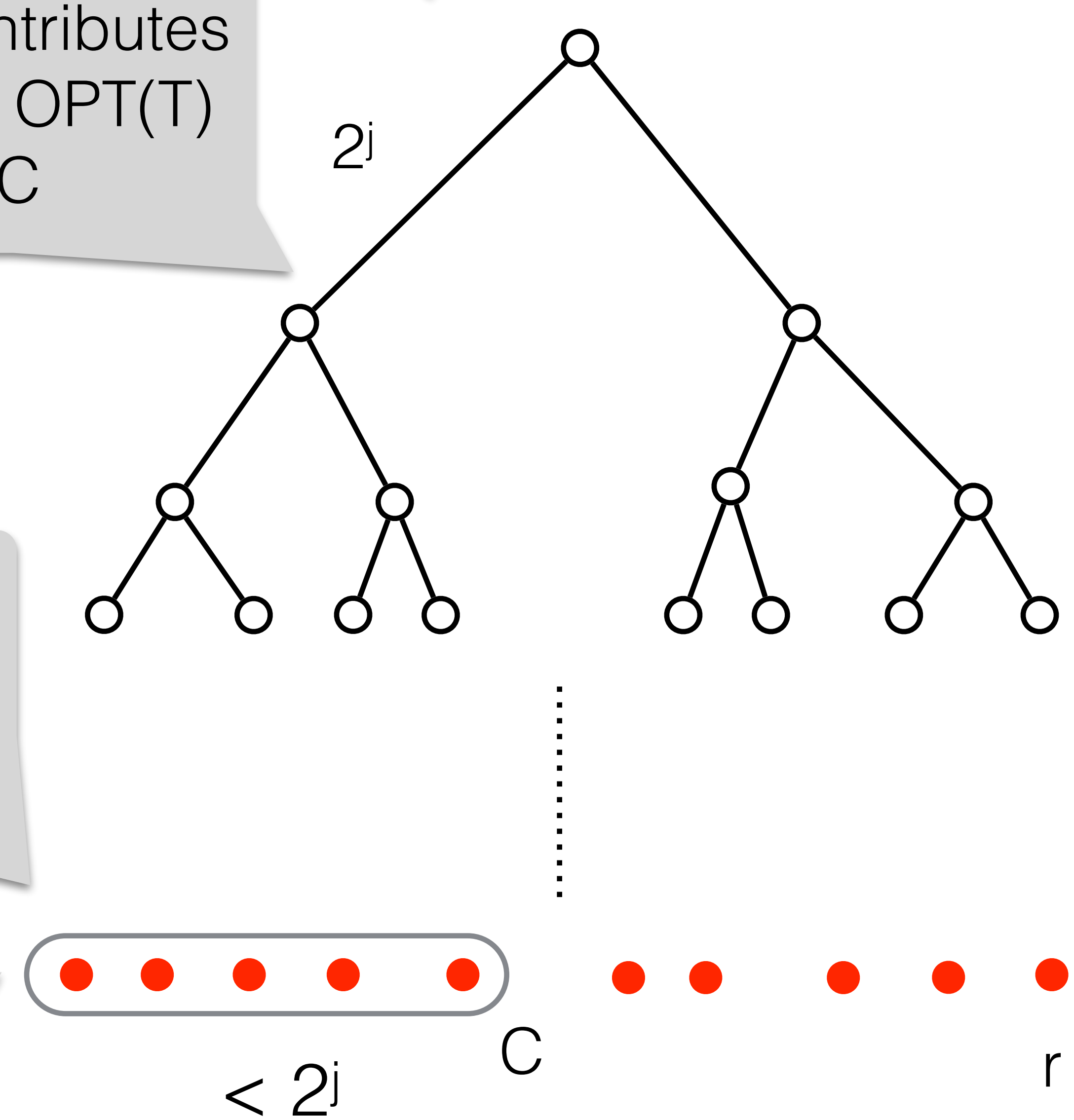
3a. Charged 2^j for each terminal in C with rent cost 2^j

1. Level-j edge contributes $2^j \min\{M, |C|\}$ to OPT(T) if r not in C

2. If terminal has rent
e rent cost
edge

Algorithm was designed with this in mind!

3b. At most M terminals in C with rent cost 2^j



Other Problems

- Works for many other “rent-or-buy”-type problems
- Connected facility location
- Prize-collecting Steiner forest (simpler analysis than [Qian-Williamson 11])

Outline

1. Overview of Analysis Framework ✔
2. Warm-Up: Steiner Tree ✔
3. Rent-or-Buy ✔
4. Steiner Network

Single-Source Steiner Network

Single-Source Steiner Network

- Given metric space and root
- Terminals with requirements R_i arrive one-by-one
- Maintain multigraph (edge-duplication allowed) s.t. terminal i has R_i edge-disjoint paths to root

Single-Source Steiner Network

- Given metric space and root
- Terminals with requirements R_i arrive one-by-one
- Maintain multigraph (edge-duplication allowed) s.t. terminal i has R_i edge-disjoint paths to root

Assume

Single-Source Steiner Network

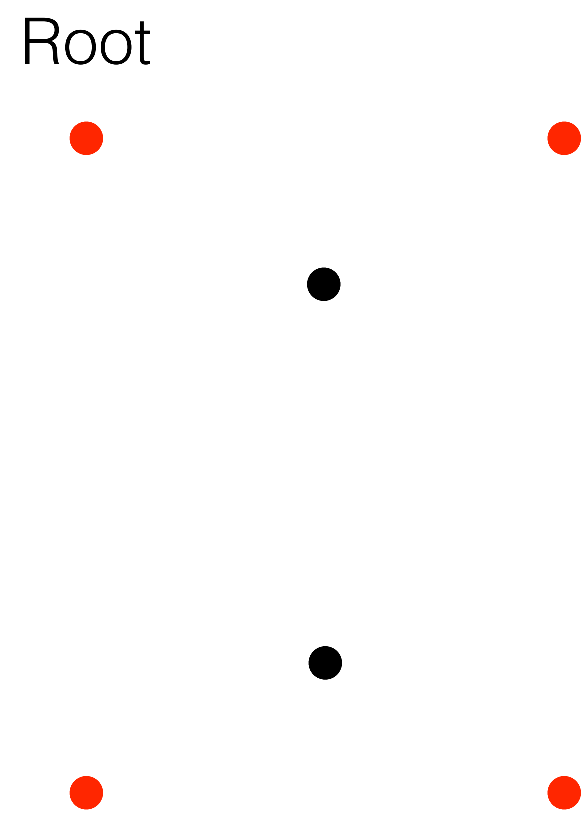
- Given metric space and root
- Terminals with requirements R_i arrive one-by-one
- Maintain multigraph (edge-duplication allowed) s.t. terminal i has R_i edge-disjoint paths to root

Assume

- Given maximum requirement (R_{max})
- Each requirement R_i is a power of 2

Intuition

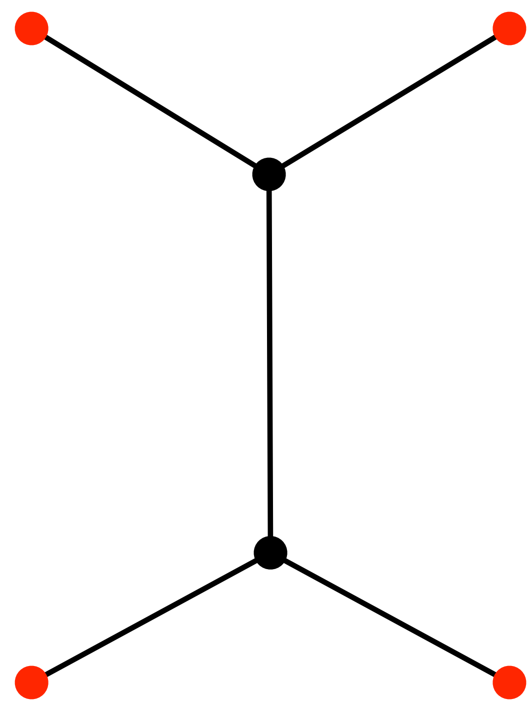
Intuition



$R_i = 2$ for all i

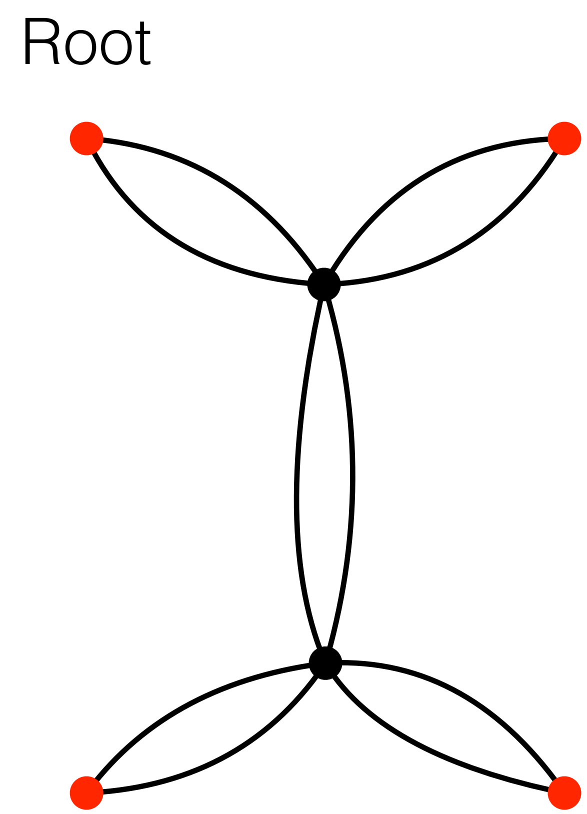
Intuition

Root



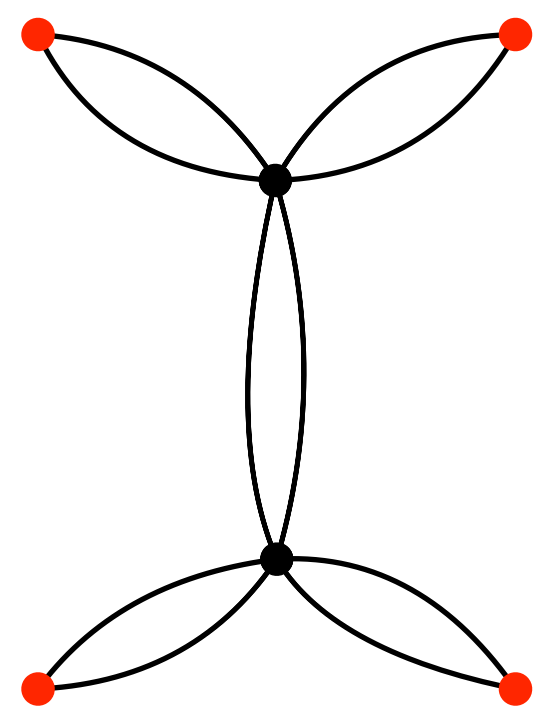
$R_i = 2$ for all i

Intuition



$R_i = 2$ for all i

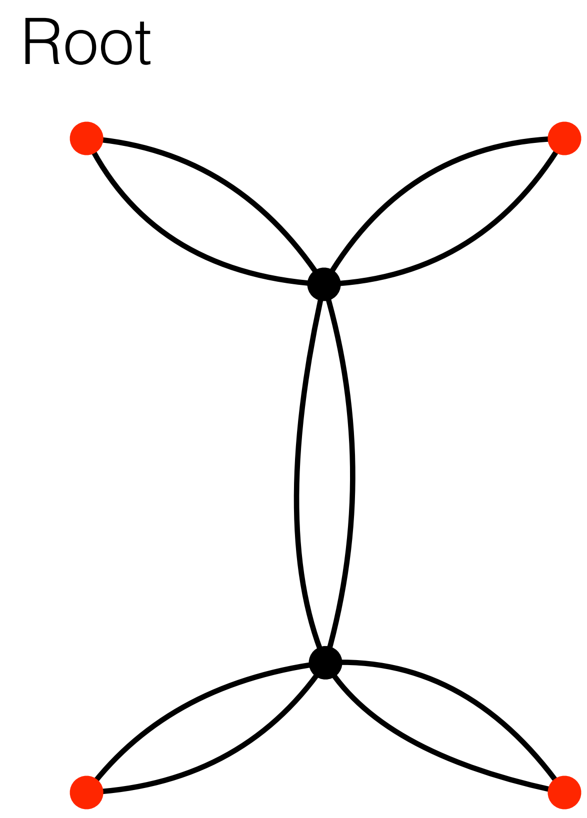
Intuition



R_i = 2 for all i

If all requirements = 2^m, then OPT = 2^m SteinerTree

Intuition



R_i = 2 for all i

If all requirements = 2^m, then OPT = 2^m SteinerTree

Key Idea: Decompose into several Steiner tree instances

Offline

Offline

[Goemans-Bertsimas 93]:

Offline

[Goemans-Bertsimas 93]:

- For each scale $0 \leq m \leq \log R_{max}$

Offline

[Goemans-Bertsimas 93]:

- For each scale $0 \leq m \leq \log R_{max}$
  - X_m = terminals with requirement 2^m and root

Offline

[Goemans-Bertsimas 93]:

- For each scale $0 \leq m \leq \log R_{max}$
  - X_m = terminals with requirement 2^m and root
  - buy 2^m copies of $MST(X_m)$

Offline

[Goemans-Bertsimas 93]:

- For each scale $0 \leq m \leq \log R_{max}$
  - X_m = terminals with requirement 2^m and root
  - buy 2^m copies of $MST(X_m)$
- $OPT \geq 2^m SteinerTree(X_m) \implies 2(\log R_{max})$-approx

Offline

[Goemans-Bertsimas 93]:

- For each scale $0 \le m \le \log R_{max}$
  - X_m = terminals with requirement 2^m and root
  - buy 2^m copies of $\mathrm{MST}(X_m)$

Tight

- $\mathrm{OPT} \ge 2^m \mathrm{SteinerTree}(X_m) \implies 2(\log R_{max})$-approx

Online

Our algorithm:

- For each scale $0 \leq m \leq \log R_{max}$
    - X_m = terminals with requirement 2^m and root
    - buy 2^m copies of Greedy(X_m)
- $\text{OPT} \geq 2^m \text{SteinerTree}(X_m) \implies O(\log k \log R_{max})$-approx

Online

Our algorithm:

- For each scale $0 \leq m \leq \log R_{max}$
    - X_m = terminals with requirement 2^m and root
    - buy 2^m copies of Greedy(X_m)
- OPT $\geq 2^m$SteinerTree(X_m) $\implies$ O(log k log R_{max})-approx

Surprisingly, our analysis gives O(log k)!

Analysis

Fix a HST embedding T

Let $OPT_m(T)$ = optimal Steiner tree for X_m in T

Analysis

Fix a HST embedding T

Subtree of T induced by X_m

Let $OPT_m(T)$ = optimal Steiner tree for X_m in T

Analysis

Fix a HST embedding T

Subtree of T induced by X_m

Let $OPT_m(T)$ = optimal Steiner tree for X_m in T

$$ALG = \sum_m 2^m \, Greedy(X_m) \leq \sum_m 2^m \, OPT_m(T)$$

Analysis

Fix a HST embedding T

Subtree of T induced by X_m

Let $OPT_m(T)$ = optimal Steiner tree for X_m in T

$$ALG = \sum_m 2^m \, Greedy(X_m) \leq \sum_m 2^m \, OPT_m(T)$$

<u>Lemma</u> $\sum_m 2^m \, OPT_m(T) \leq 2 \, OPT(T)$

Analysis

Fix a HST embedding T

Subtree of T induced by X_m

Let $OPT_m(T)$ = optimal Steiner tree for X_m in T

$$ALG = \sum_m 2^m \, Greedy(X_m) \leq \sum_m 2^m \, OPT_m(T)$$

<u>Lemma</u> $\sum_m 2^m \, OPT_m(T) \leq 2 \, OPT(T)$

On trees, optimal SS Steiner network solution decomposes into log R_{max} optimal Steiner tree solutions at a loss of 2.

Analysis

Fix a HST embedding T

Subtree of T induced by X_m

Let $OPT_m(T)$ = optimal Steiner tree for X_m in T

$$ALG = \sum_m 2^m \, Greedy(X_m) \leq \sum_m 2^m \, OPT_m(T)$$

<u>Lemma</u> $\sum_m 2^m \, OPT_m(T) \leq 2 \, OPT(T)$

On trees, optimal SS Steiner network solution decomposes into log R_{max} optimal Steiner tree solutions at a loss of 2.

<u>Theorem</u>:
ALG is O(log k)-competitive for SS Steiner network

Summary

Summary

- New analysis framework based on HST embeddings and metric decompositions

Summary

- New analysis framework based on HST embeddings and metric decompositions

- Unified approach for online network design

Summary

- New analysis framework based on HST embeddings and metric decompositions

- Unified approach for online network design

- Question: for what other problems can we apply this framework to?

Summary

- New analysis framework based on HST embeddings and metric decompositions
- Unified approach for online network design
- Question: for what other problems can we apply this framework to?
- Progress so far: multicast games, spanners, LASTs, simultaneous buy-at-bulk, Steiner leasing [Bienkowski-Kraska-Schmidt 17]

Summary

- New analysis framework based on HST embeddings and metric decompositions
- Unified approach for online network design
- Question: for what other problems can we apply this framework to?
- Progress so far: multicast games, spanners, LASTs, simultaneous buy-at-bulk, Steiner leasing [Bienkowski-Kraska-Schmidt 17]

Thank you!